Universidad de Alcalá
Escuela Politécnica Superior

Máster Universitario en Ingeniería de Telecomunicación

Trabajo Fin de Máster

Sistema de posicionamiento óptimo de micrófonos en entornos cerrados basado en optimización multi-objetivo para tareas de localización

Autor: Roberto Macho Pedroso

Tutora: Cristina Losada Gutiérrez

2017

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Máster Universitario en Ingeniería de Telecomunicación

Trabajo Fin de Máster

Sistema de posicionamiento óptimo de micrófonos en entornos cerrados basado en optimización multi-objetivo para tareas de localización

Autor: Roberto Macho Pedroso

Tutora: Cristina Losada Gutiérrez

Tribunal:

Presidente: Manuel Rosa Zurera

Vocal 1º: Javier Macías Guarasa

Vocal 2º: Cristina Losada Gutiérrez

Calificación: .....

Fecha: .....

A todos los que han estado a mi lado a lo largo de mi vida...

Agradecimientos

El agradecimiento es la memoria del corazón.

Lao Tse

Por fin ha llegado el momento de terminar el máster, y por el momento mi etapa universitaria.

En primer lugar a mi tutora Cristina por su ayuda y paciencia.

A Fran como tutor de parte de los trabajos previos.

A Javi, Jose y los demás miembros del GEINTRA que tanto me han ayudado.

Al resto de profesores de la EPS que me han enseñado todo lo que sé.

A mis padres, Jesús y Conchi, por apoyarme en lo necesario para terminar mi carrera.

A mis hermanos, Héctor y Aurora, que con su apoyo me han animado cuando lo he necesitado.

A mis amigos que han sabido aguantarme todos estos años y me han insistido para terminar.

Por último y no menos importante a Sabrina por hacerme más llevadera la vida desde que la conozco y por saber sacarme siempre una sonrisa incluso cuando yo creo que no es posible.

A todos ellos, gracias por hacerme llegar hasta aquí.

Resumen

En este trabajo se ha implementado y evaluado en Matlab un sistema de posicionamiento óptimo de sensores, genéricos o micrófonos, para la localización de objetivos en un entorno cerrado.

La ubicación óptima se obtiene mediante un algoritmo genético, mono-objetivo o multi-objetivo, diseñado e implementado en este trabajo, que minimiza el error de localización para sistemas basados en TDOA o SRP, utilizando el modelo acústico de fuente-imagen con y sin obstáculos.

Todo el sistema está gobernado con una GUI de Matlab que permite agilizar el uso de las funciones y modificar fácilmente los parámetros.

Para la validación del sistema se ha llevado a cabo una experimentación exhaustiva en entornos simulados en Matlab.

Palabras clave: Localización, SRP, posicionamiento óptimo de sensores, agrupaciones de micrófonos, optimización multi-objetivo.

Abstract

This work outlines the features and functioning of a specifically created MATLAB-developed system which allows obtaining the optimal positioning of sensors or microphone arrays for the localization of objectives in a closed environment.

This optimal positioning is obtained thanks to a mono-objective or multi-objective genetic algorithm that minimizes localization errors for TDOA or SRP-based systems, using the image-source acoustic model with or without obstacles.

The whole system is being ruled by a MATLAB graphic interface which allows to accelerate the usage of functions and easily modify all of their parameters.

To validate the system a complete experimentation has been carried out in Matlab simulated environments.

Keywords: Localization, SRP, optimal positioning of sensors, microphone arrays, multi-objective optimization.

Resumen extendido

La localización y el seguimiento de objetivos (persona/s u objeto/s) en un espacio inteligente es fundamental para mejorar los procesos de interacción con el entorno.

El principal problema de la localización en entornos cerrados es la limitación de técnicas que encontramos, ya que no podemos usar técnicas como la localización por GPS o radiobalizas. Por eso en espacios cerrados, hay que hacer uso de otro tipo de sensores como infrarrojos, cámaras, micrófonos...

En este trabajo se ha optado por el uso de micrófonos para la localización, ya que no es necesario utilizar dispositivos activos en el objetivo (como los sensores infrarrojos), ni invade la privacidad (como las cámaras), por lo que es menos invasivo.

Para comenzar, en este trabajo se realiza un estudio teórico de los distintos métodos de localización permitidos por el sistema desarrollado, mediante algoritmos basados en Time Diference of Arrival (TDOA) y Steered Response Power (SRP) (TDOA para todo tipo de sensores y SRP únicamente para micrófonos).

Posteriormente se define el modelo acústico simulado, basado en el modelo SRP apoyado con el método de fuente imagen que permite calcular los rebotes, para calcular el error de localización.

El estudio teórico sigue con la forma de minimizar ese error mediante el uso de algoritmos metaheurísticos, en concreto los genéticos mono-objetivo y multi-objetivo. Esta técnica permite reducir considerablemente el tiempo de cálculo de las ubicaciones óptimas de los sensores o micrófonos, ya que se basa en recrear los procesos de evolución de la naturaleza mediante cruces y mutaciones de las distintas soluciones. Al no ser un método determinista, ya que no se prueban las infinitas soluciones posibles, no nos proporciona un resultado absoluto pero si nos proporciona una solución aceptable en un tiempo razonable.

En este trabajo se plantea un sistema desarrollado en MATLAB que permite obtener el posicionamiento óptimo de un conjunto de sensores para la localización de objetivos en un entorno cerrado, en base al error de localización.

Para ello, el sistema desarrollado calcula el error de localización a partir de la probabilidad de que un objetivo determinado este en cada localización posible, mediante una SLF derivada del patrón SRP-PHAT obtenido mediante un modelo simulado. Además en este punto tiene en cuenta los obstáculos de la sala que se quieran tener en cuenta, ya que previo al cálculo del patrón SRP se verifica la visibilidad de cada localización posible y cada punto imagen para cada micrófono, de forma rápida ya que previamente se han calculado las zonas ciegas para cada localización y punto imagen.

Una vez obtenido el error de localización el sistema desarrollado calcula el posicionamiento óptimo de sensores mediante un algortimo genético mono-objetivo o multi-objetivo diseñado para este problema, con la mayor variedad posible de funciones de mutación y cruce, con la posibilidad de corrección. El criterio de selección del algoritmo mono-objetivo se ha realizado por ranking y el del multi-objetivo se ha realizado siguiendo el modelos de NGSA-II.

Todo el sistema está gobernado con una interfaz gráfica de MATLAB (GUI) que permite agilizar el uso de las funciones y modificar muy fácilmente todos los parámetros de éstas.

Esta interfaz gráfica permite modificar la información a tener en cuenta de la sala (tipo de cómputo de obstáculos), el tamaño de rejilla de puntos objetivo, realizar modificaciones de la topología de los arrays de micrófonos, el número de sensores a utilizar y las posiciones posibles de éstos, las funciones utilizadas por el algoritmo genético,...

Además permite no sólo ver la simulación al acabar el resultado, sino también ver resultados anteriores para poder analizar los distintos métodos de análisis del sistema. También está desarrollado con el fin de poder ver también el funcionamiento interno del algoritmo genético, ya que esta diseñado para almacenar los resultados intermedios, con la posibilidad de ver el funcionamiento del algoritmo en vivo.

Por último se ha hecho una experimentación exhaustiva del funcionamiento del sistema desarrollado, tanto en su versión mono-objetivo como multi-objetivo, para un amplio conjunto de salas y con configuraciones diferentes de sensores.

Con el análisis cuantitativo de los experimentos realizados se ha demostrado el funcionamiento correcto del sistema implementado y por último se presentan próximas líneas futuras de desarrollo para un mayor aprovechamiento del sistema desarrollado.

Índice general

Índice de figuras

Índice de tablas

Lista de acrónimos

A-GPS	Assisted Global Positioning System.
BEM	Boundary Element Method.
CRLB	Cramér-Rao lower bound.
DOA	Direction of Arrival.
FDTD	Finite-Different Time-Domain.
FEM	Finite Element Method.
FIM	Fisher Information Matrix.
GA	Genetic Algorithm.
GCC-PHAT	Generalized cross-correlations PHAT.
GEINTRA	Grupo de Ingeniería Electrónica aplicada a Espacios Inteligentes y Transporte.
GPS	Global Positioning System.
GUI	Interfaz Gráfica de Usuario.
IR	Infrarrojos.
ISM	Image-Source Method.
LOS	Línea de visión.
NSGA	Nondominated Sorting Genetic Algorithm.
NSGA-II	Nondominated Sorting Genetic Algorithm II.
PAES	Pareto Archived Evolution Strategy.
PHAT	Transformación de fase.
RGB	Red, Green, Blue.
RSS	Received Signal Strength.
RW-GA	Random Weights Genetic Algorithm.

SEA	Static Energy Analysis.
SLF	Spatial Likelihood Function.
SPEA	Strength Pareto Evolutionary Algorithm 2.
SPEA	Strength Pareto Evolutionary Algorithm.
SRP	Steered Response Power.
TDOA	Time Difference of Arrival.
TFM	Trabajo Fin de Máster.
TOA	Time of Arrival.
VEGA	Vector Evaluated Genetic Algorithm.
VOW-GA	Variable Objective Weighting Genetic Algorithm.

Capítulo 1

Introducción

El objetivo de una buena introducción definitiva es que el lector se contente con ella, lo entienda todo y no lea el resto

Como se hace una tesis. Umberto Eco

1.1 Introducción

El análisis automático de espacios inteligentes a partir del procesamiento de múltiples sensores es un área de cada vez mayor actividad científica. En ese contexto, las tareas de detección, localización y seguimiento de objetivos (personas o cualquier otro elemento móvil) son fundamentales para mejorar los procesos de interacción con el entorno, o con otras personas u objetos dentro del mismo.

El Grupo de Ingeniería Electrónica aplicada a Espacios Inteligentes y Transporte (GEINTRA) del Departamento de Electrónica de la Universidad de Alcalá ha iniciado una línea de actividad en la que se han ido planteando trabajos orientados a la generación de tecnología basada en el procesamiento de las señales captadas por agrupaciones de sensores de distintos tipos: señales acústicas, tanto en la banda de audio como en la de ultrasonidos, señales infrarrojas, y señales de vídeo con cámaras de distintos tipos (Red, Green, Blue (RGB), de tiempo de vuelo, etc.)

Una etapa fundamental en el despliegue de agrupaciones de sensores en un entorno real es la de la decisión acerca de la topología y distribución espacial de todos ellos. El enfoque tradicional asume una topología conocida a priori, y una distribución geométrica que trata de cubrir razonablemente el espacio objeto de análisis. El inconveniente fundamental de esta estrategia es que, en general, no utiliza criterios objetivos y medibles para la toma de decisiones, lo que puede implicar limitaciones en las prestaciones que finalmente se obtendrán de los algoritmos que usan esos sensores.

Una alternativa deseable sería la determinación automática de dicha topología y distribución espacial, utilizando algoritmos de optimización basados en distintos paradigmas y que se basan en el uso de métricas objetivas sobre las que se construyen las funciones a optimizar [6, 7].

Esta tarea es necesaria para mejorar el rendimiento en entornos complejos de escenarios con muchos sensores. Esto es necesario en aplicaciones de posicionamiento automático de sensores en edificios [8], vídeo vigilancia [9], tareas generales de visión [10], visión robótica [11], localización de robots [12], localización de fuente acústica [13], además de otras técnicas, que pueden verse beneficiadas del posicionamiento óptimo de sensores.

Una de estas técnicas de optimización son los algoritmos genéticos [14], que se basan en una búsqueda heurística que imita el proceso de la selección natural. Dicho proceso se basa en aplicar los métodos de cruce, mutación y selección que tienen lugar en la naturaleza a una población de soluciones con el fin de encontrar la mejor solución posible.

El Trabajo Fin de Máster (TFM) realizado se centra en la continuación de la línea de investigación iniciada en con los trabajos previos [15] y [16], donde se establece el punto de partida para este trabajo. En dichos trabajos se pone de manifiesto la necesidad de la utilización de algoritmos genéticos multi-objetivo, ya que al aplicar distintas métricas al problema de optimización se generan soluciones muy distintas.

En este trabajo se aborda de forma más exhaustiva el problema de estimación de la ubicación óptima de micrófonos (individuales o formando agrupaciones (*arrays*)), en escenarios arbitrarios, para la localización acústica. Para ello se utilizará el modelo acústico [17] realizado recientemente y se apoyará en el estado del arte de las estrategias de optimización evolutiva [18].

1.2 Objetivos

El objetivo fundamental de este trabajo es el estudio, implementación y evaluación de un sistema que permita estimar la ubicación óptima de un número determinado de micrófonos (definidos de forma individual o en agrupaciones (*arrays*) con una geometría previamente definida), en entornos cerrados arbitrarios, usando técnicas de optimización evolutiva y considerando múltiples objetivos simultáneos.

Los objetivos específicos de este trabajo son los siguientes:

- Realizar un estudio de las alternativas del estado del arte para el posicionamiento óptimo multi-objetivo de micrófonos en entornos cerrados, y los modelos de propagación acústica utilizables.
- Realizar la implementación del sistema en entorno Matlab, que incluirá tanto los procesos de optimización, como los de generación de datos de propagación acústica en el entorno (en los que se basan los modelos probabilísticos de cálculo de errores de localización que servirán de base a la definición de objetivos de optimización), incluyendo el tratamiento de obstáculos en las salas.
- Realizar una evaluación exhaustiva y rigurosa de la algorítmica implementada mediante simulación de distintos entornos en un conjunto amplio de condiciones experimentales.
- Además tras los trabajos anteriores a éste se observó como el uso de las librerías de algoritmos genéticos definidas en Matlab, no funcionaban de forma correcta para el problema a tratar por lo también se requiere el desarrollo de nuevas funciones para el uso de algoritmos genéticos.

1.3 Estructura de la memoria

A continuación se describe la organización y la estructura de cada uno de los capítulos contenidos en esta memoria.

En el Capítulo 2 se realiza el estudio teórico de las distintas técnicas de localización usadas en este trabajo, de los modelos de simulación acústica y del algoritmo, tanto mono-objetivo como multi-objetivo, de posicionamiento óptima de los sensores.

A lo largo del Capítulo 3 se explica detalladamente el sistema desarrollado en este trabajo, su configuración y su funcionamiento interno.

En el Capítulo 4 se hace un análisis cualitativo y cuantitativo exhaustivo del funcionamiento del sistema desarrollado, tanto en con el modelo mono-objetivo rediseñado, como con el modelo multi-objetivo.

El Capítulo 5 recoge las principales conclusiones y líneas de trabajo futuro.

En el Capítulo 6 se detallan los requisitos para la utilización del sistema desarrollado.

El Capítulo 7 contiene el presupuesto detallado de este proyecto.

Capítulo 2

Estado del arte

El conocimiento comienza por la práctica y todo
conocimiento teórico, adquirido a través de la práctica,
debe volver a la práctica.

Sobre la práctica y la contradicción. Mao Zedong

2.1 Introducción

En este capítulo se aborda el estado del arte de los sistemas de posicionamiento de sensores o micrófonos mediante algoritmos genéticos apoyado fuertemente por un estudio teórico de las técnicas necesarias para su realización haciendo hincapié en el estado del arte de cada una de las técnicas.

En primer lugar se analizan los métodos de localización en entornos cerrados, profundizando en los métodos basados en TDOA (para sensores genéricos) y SRP (para micrófonos). Se describe el problema de localización utilizando micrófonos, analizando los modelos acústicos que se pueden emplear para su simulación, haciendo énfasis en el método de fuente-imagen para utilizarlo posteriormente en el modelo basado en SRP-PHAT para obtener el error de localización de forma simulada. Por último se analiza el problema de posicionamiento óptimo de los sensores desarrollando los algoritmos genéticos mono-objetivo y multi-objetivo.

Como ya se ha comentado, este trabajo se enmarca dentro de una de las líneas de investigación del grupo GEINTRA y toma como punto de partida los trabajos previos descritos en [15] y [16]. Por este motivo, una parte del contenido de este capítulo se ha adaptado a partir de los trabajos citados.

2.2 Introducción al problema de localización

El objetivo de la localización en interiores es obtener, con la mejor precisión posible, la posición de un objetivo que emita una señal dentro de un habitáculo o recinto.

Para ello es necesario el uso de sensores, ya sean infrarrojos, ultrasonidos, micrófonos, etc. dependiendo del tipo de señal que emita el objetivo. Empleando las señales medidas por dichos sensores, podemos obtener información de la posición, procesando las medidas con un ordenador personal o un microprocesador para aplicaciones más reducidas.

Dicha localización se hace de forma pasiva ya que el objetivo únicamente emite una señal pero no hace ningún procesamiento. Este proceso puede ser reversible y ser el objetivo, el sensor que a partir de la

información de los emisores fijos, estime su posición de forma activa, de forma análoga al *Assisted Global Positioning System (A-GPS)* que utiliza triangulación de señales de telefonía que llegan al teléfono [19].

Los métodos de localización pueden dividirse en dos grandes grupos en función de cómo se realiza el cálculo de la posición, tal como se describe a continuación.

2.2.1 Métodos basados en el tiempo, la potencia o la dirección de llegada de la señal

Este apartado ha sido adaptado de [15].

Este tipo de métodos se usan para calcular la posición a partir de la información recibida de las señales electromagnéticas u ondas acústicas.

Este cálculo de la posición se puede realizar en función del tiempo de llegada (*Time of Arrival (TOA)* [20]), la diferencia de tiempo de llegada (*Time Difference of Arrival (TDOA)* [20, 21]), la potencia de la señal recibida (*Received Signal Strength (RSS)* [22]) o la dirección de llegada (*Direction of Arrival (DOA)* [23]) de las señales recibidas por los sensores. TOA, TDOA y RSS proporcionan la información de la posición a partir de la distancia entre la fuente y los sensores, mientras que en DOA la orientación de la fuente es relativa a los sensores.

En la Tabla 2.1 se puede ver un resumen de las características de estos métodos, sus ventajas y sus inconvenientes.

La propagación de la Línea de visión (LOS) se refiere a la radiación de ondas electromagnéticas o de ondas acústicas. Implica que tiene que haber un camino directo entre el emisor y el receptor.

Tabla 2.1: Comparación entre los diferentes métodos de medida (adaptada de [5])

Método	**Información de la Localización**	**Ventajas**	**Inconvenientes**
TOA	Distancia	Alta precisión	Es necesaria la sincronización temporal tanto en la fuente como en todos los sensores. LOS necesaria
TDOA	Diferencia de distancias	Alta precisión No es necesaria la sincronización temporal en la fuente	LOS necesaria
RSS	Distancia	Simple y económico No es necesaria la sincronización temporal en la fuente	Baja precisión
DOA	Orientación	Sólo son necesarios dos sensores. No es necesaria la sincronización temporal	Son necesarias antenas inteligentes LOS necesaria

Teniendo esto en cuenta, el resto del trabajo se centrará en el uso del método TDOA, ya que ofrece alta precisión eliminando la necesidad de sincronización con la fuente. Este método utiliza uno de los sensores como referencia y calcula la diferencia de tiempo de llegada de las señales a los otros sensores con respecto al de referencia.

En la Figura 2.1 se puede ver un ejemplo de funcionamiento del método TDOA en dos dimensiones en una habitación con cuatro sensores. En la Figura se puede observar cómo usando un algoritmo basado en TDOA se realiza la localización mediante trilateración hiperbólica [24].

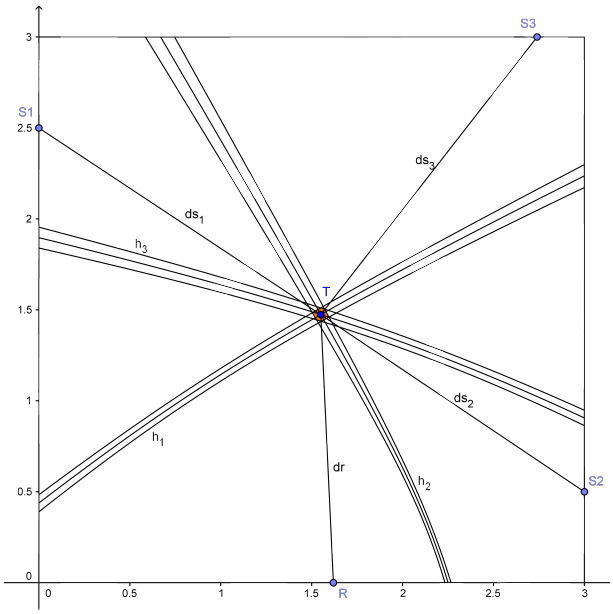


Figura 2.1: TDOA con 4 sensores en 2D

La localización se realiza trazando $N-1$ cónicas (hipérbolas), siendo N el número de sensores y la hipérbola el lugar geométrico de los puntos del plano que mantiene constante la diferencia de la distancia a los dos focos, siendo dichos focos el sensor de referencia y cada uno de los $N-1$ sensores respectivamente, que pasa por el objetivo T.

En ausencia de ruido dichas hipérbolas se cortarían en un único punto, pero teniendo el ruido presente aparecen dos hipérbolas más para cada pareja de sensores, produciéndose una región de estimación.

Para la localización en dos dimensiones son necesarios al menos 3 sensores (uno de referencia y dos más para calcular la diferencia de distancias), sin embargo para la localización en el espacio tridimensional es necesario el uso de al menos 4 sensores, empleándose hiperboloides en lugar de hipérbolas.

Una vez obtenido el tiempo de llegada de la señal a los N sensores, se selecciona uno de ellos como referencia y se calcula la diferencia de medida entre cada uno de los otros $N-1$ sensores y el de referencia [25]. Las $N-1$ diferencias de medidas pueden ser expresadas como una función de los parámetros a estimar ($\theta = [x\ y\ z]^T$):

$$r = \mu(\theta) + \epsilon \tag{2.1}$$

donde cada elemento es un vector columna de tamaño $N-1$ con la forma mostrada en la ecuación 2.2.

$$r = \begin{pmatrix} \Delta d_{1,r} \\ \Delta d_{2,r} \\ \vdots \\ \Delta d_{N-1,r} \end{pmatrix}; \epsilon = \begin{pmatrix} \epsilon_{1,r} \\ \epsilon_{2,r} \\ \vdots \\ \epsilon_{N-1,r} \end{pmatrix}; \mu(\theta) = \begin{pmatrix} \|\theta - x_1\| - \|\theta - x_r\| \\ \|\theta - x_2\| - \|\theta - x_r\| \\ \vdots \\ \|\theta - x_{N-1}\| - \|\theta - x_r\| \end{pmatrix} \tag{2.2}$$

Asumiendo que sigue una distribución normal, la diferencia de medidas se puede modelar como $r \sim \mathcal{N}(\mu(\theta), \Sigma)$. Siendo ϵ el error en cada medida (con una media nula y varianza $\sigma^2_{x,r}$), μ el vector de las diferencia de normas (distancias) de los sensores al objetivo y la referencia al objetivo y $\Delta d_{x,r}$ la diferencia de medida entre el sensor x y el de referencia.

La matriz de covarianza es una matriz cuadrada de orden $N-1$ formada por las varianzas y las covarianzas de las variables, que modela el error de los sensores para un método de medida en concreto.

Para el método TDOA, que es en el que se centra este trabajo, la matriz de covarianza es de la forma de la ecuación 2.3. Como el error de la referencia está presente en cada medida hay correlación entre éstas, por lo cual los elementos de fuera de la diagonal principal en la matriz de covarianza no son nulos.

$$\Sigma = \begin{pmatrix} \sigma_{1,r}^2 & \sigma_r^2 & \cdots & \sigma_r^2 \\ \sigma_r^2 & \sigma_{2,r}^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \sigma_r^2 \\ \sigma_r^2 & \cdots & \sigma_r^2 & \sigma_{N-1,r}^2 \end{pmatrix} \tag{2.3}$$

siendo:

$$\sigma_{x,r}^2 = \sigma_x^2 + \sigma_r^2 \tag{2.4}$$

El cálculo de la matriz de covarianza se realiza teniendo en cuenta que el error de los sensores es dependiente de la distancia de éstos al objetivo. Dicha matriz será una matriz simétrica de tamaño $N-1$ tal como se muestra en la ecuación 2.5.

$$\Sigma = \begin{pmatrix} dr^2 + ds_1^2 & dr^2 & \cdots & dr^2 \\ dr^2 & dr^2 + ds_2^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & dr^2 \\ dr^2 & \cdots & dr^2 & dr^2 + ds_{N-1}^2 \end{pmatrix} \tag{2.5}$$

De forma simplificada se puede calcular suponiendo que el error es independiente de la distancia entre el emisor y el sensor, con unos en su diagonal y 0.5 en el resto (ecuación 2.6).

$$\Sigma = \begin{pmatrix} 1 & 0{,}5 & \cdots & 0{,}5 \\ 0{,}5 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0{,}5 \\ 0{,}5 & \cdots & 0{,}5 & 1 \end{pmatrix} \tag{2.6}$$

Para resolver el problema hay que utilizar algún estimador no lineal. Se sabe que existen estimadores eficientes o asintóticamente eficientes (como máxima verosimilitud). Un estimador es eficiente si la varianza de la estimación alcanza la cota inferior de Cramér-Rao (*Cramér-Rao lower bound (CRLB)*) [26], así que podemos utilizarla como función objetivo a minimizar para encontrar un conjunto de sensores que ofrezcan el menor error en la posición estimada posible.

La CRLB permite obtener una cota inferior de la varianza de un estimador insesgado de un parámetro determinista. Ésta se puede calcular con la ecuación 2.7. Esta función se obtiene calculando la inversa de la Matriz de Información de Fisher (*Fisher Information Matrix (FIM)*), la cual se obtiene con la segunda derivada de la función de verosimilitud. En algunos casos esa cota no existe dando como resultado infinito, cuando la FIM es singular.

$$CRLB = [J^T \cdot \Sigma^{-1} \cdot J]^{-1} \tag{2.7}$$

siendo J el jacobiano de la función $\mu(\theta)$ definido en la ecuación 2.2. Así, la fórmula del jacobiano para TDOA es la mostrada en 2.8.

$$J = \begin{pmatrix} \left(\frac{x_T - x_{S1}}{ds_1} - \frac{x_T - x_R}{dr}\right) & \left(\frac{y_T - y_{S1}}{ds_1} - \frac{y_T - y_R}{dr}\right) & \left(\frac{z_T - z_{S1}}{ds_1} - \frac{z_T - z_R}{dr}\right) \\ \left(\frac{x_T - x_{S2}}{ds_2} - \frac{x_T - x_R}{dr}\right) & \left(\frac{y_T - y_{S2}}{ds_2} - \frac{y_T - y_R}{dr}\right) & \left(\frac{z_T - z_{S2}}{ds_2} - \frac{z_T - z_R}{dr}\right) \\ \vdots & \vdots & \vdots \\ \left(\frac{x_T - x_{S(N-1)}}{ds_{N-1}} - \frac{x_T - x_R}{dr}\right) & \left(\frac{y_T - y_{S(N-1)}}{ds_{N-1}} - \frac{y_T - y_R}{dr}\right) & \left(\frac{z_T - z_{S(N-1)}}{ds_{N-1}} - \frac{z_T - z_R}{dr}\right) \end{pmatrix} \tag{2.8}$$

De esta forma se tiene en cuenta tanto el error en las medidas, con la información contenida en la matriz de covarianza, como la geometría de la sala, pues cada fila es una diferencia de vectores unitarios que indican la dirección $\overrightarrow{TS}$ y $\overrightarrow{TR}$. Por lo que optimizando la CRLB se obtiene un conjunto de sensores que consigue un menor error cuadrático medio en el posicionamiento [27].

2.2.2 Métodos basados en *Steered Response Power (SRP)*

Este apartado ha sido adaptado de [15] y [16].

A diferencia de los métodos basados en el tiempo de llegada (TOA), los métodos basados en *beamforming* (o conformación de haces) sólo pueden utilizarse para localización acústica ya que hacen uso de la capacidad de los micrófonos de dirigir el patrón de actividad a distintas direcciones del espacio.

Los métodos basados en SRP hacen uso de técnicas *beamforming* con el objetivo estimar la localización de la fuente maximizando o minimizando una estadística espacial asociada con cada posición. En la aproximación SRP, que es el método de conformación de haces más sencillo, la estadística se basa en la potencia de señal recibida cuando el conjunto de micrófonos es dirigido en la dirección de una ubicación específica. Por lo tanto, se supone que la posición de la fuente es consistente con la posición correspondiente a la potencia de señal máxima estimada.

Combinando el método SRP con la transformación de fase (Transformación de fase (PHAT)) nace SRP-PHAT, que se propuso por primera vez en [28]. Éste es un algoritmo ampliamente utilizado para la localización de locutores basado en *beamforming*, que combina la robustez de los métodos de conformación de haces dirigidos con la insensibilidad a las condiciones de señal proporcionadas por el filtrado PHAT.

El formador de haz de retardo y suma clásico utilizado en SRP se reemplaza en SRP-PHAT por un formador de haz de filtro y suma utilizando el filtrado PHAT para ponderar las señales entrantes. El clásico *delay-and-sum beamformer* utilizado en SRP se remplaza en SRP-PHAT con un *filter-and-sum beamformer*, utilizando el filtrado PHAT para ponderar las señales entrantes. En este trabajo ya que sólo se hará uso del método SRP-PHAT el término SRP se utilizará indistintamente con SRP-PHAT.

La ventaja de utilizar PHAT es que no se hacen suposiciones sobre la señal o las condiciones de habitación, y esta es la razón de la robustez del método SRP-PHAT en escenarios reverberantes, donde la localización de la fuente es desconocida. Este método se define generalmente como un estándar para la localización de la fuente acústica, siendo un algoritmo ampliamente utilizado para la localización del locutor.

El mapa de potencia SRP-PHAT es una representación adecuada para la localización de fuentes acústicas cuando sólo hay una única fuente activa y se emplean suficientes micrófonos. De lo contrario, la redundancia inherente de SRP-PHAT hace realmente difícil separar las fuentes y distinguir entre fuentes y artefactos. Sin embargo en escenarios reales, donde sólo unas pocas fuentes están activas al mismo tiempo y la mayor parte del espacio está vacía, la representación simple que proporciona SRP-PHAT puede ser muy útil.

Para calcular ese mapa de potencia SRP-PHAT, se tiene que contar con un entorno equipado con un conjunto de N_μ micrófonos $\mathcal{M} = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_{N_\mu}\}$, donde $\mathbf{m}_i$ es un vector tridimensional $\mathbf{m}_i = (m_{ix}, m_{iy}, m_{iz})^\top$ que denota la posición de cada micrófono desde la coordinada de origen de referencia.

Con esta configuración si se asume que existe una fuente acústica localizada en la posición genérica $\mathbf{r} = (r_x, r_y, r_z)^\top$, emitiendo una señal acústica $x(t)$. Denominando $x_i(t)$ a la señal recibida en un micrófono localizado en $\mathbf{m}_i$, y $\mathbf{q} = (q_x, q_y, q_z)^\top$ a la localización de un objetivo genérico a la cual se quiere dirigir el conjunto de micrófonos. Usualmente se discretiza el espacio usando un conjunto finito, $\mathcal{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_Q\}$, de Q vectores tridimensionales.

El *delay-and-sum beamformer* alinea el conjunto de señales $x_1(t), \cdots, x_{N_\mu}(t)$, compensando los retardos de propagación desde la posición de destino $\mathbf{q}$ a cada micrófono $\mathbf{m}_i$. La señal *beamformed* resultante cuando la matriz es dirigida a $\mathbf{q}$ se define empleando la ecuación 2.9 (incluyendo también la expresión de dominio de frecuencia):

$$y(t, \mathbf{q}) = \sum_{i=1}^{N_\mu} x_i\left(t + \tau_i(\mathbf{q})\right) \overset{\mathcal{F}}{\longleftrightarrow} Y(\omega, \mathbf{q}) = \sum_{i=1}^{N_\mu} X_i(\omega) \mathrm{e}^{\mathrm{j}\omega\tau_i(\mathbf{q})} \quad , \tag{2.9}$$

donde $X_i(\omega)$ es la Transformada de Fourier de $x_i(t)$, y $\tau_i(\mathbf{q})$ es el retraso de propagación entre $\mathbf{m}_i$ y $\mathbf{q}$, que es calculado como $\tau_i(\mathbf{q}) = \frac{1}{c}\|\mathbf{q} - \mathbf{m}_i\|$, siendo c la velocidad del sonido en el aire.

El *filter-and-sum beamformer* es una generalización del *delay-and-sum beamformer*, donde se aplica un filtrado adaptativo a las señales de los micrófonos. En este caso, la señal recibida en el micrófono $\mathbf{m}_i$ es filtrada con $H_i(\omega)$.

La señal *beamformed* donde el conjunto de micrófonos es dirigido a la posición $\mathbf{q}$ es definida en el dominio de la frecuencia por:

$$Y(\omega, \mathbf{q}) = \sum_{i=1}^{N_\mu} H_i(\omega)\, X_i(\omega)\, \mathrm{e}^{\mathrm{j}\omega\tau_i(\mathbf{q})} \quad . \tag{2.10}$$

La SRP puede ser expresada como la potencia de salida de la señal recibida por un *filter-and-sum beamformer* de N_μ elementos [29]:

$$P(\mathbf{q}) = \int_{-\infty}^{-\infty} |Y(\omega, \mathbf{q})|^2 \, \mathrm{d}\omega \quad , \tag{2.11}$$

donde $P(\mathbf{q})$ es la potencia recibida en la posición $\mathbf{q}$.

Por tanto SRP-PHAT es un caso particular de SRP donde $H_i(\omega) = |X_i(\omega)|^{-1}$. La ponderación PHAT reduce empíricamente la influencia de la trayectoria múltiple en el SRP-PHAT de una señal que llega a dos micrófonos diferentes en un entorno acústico reverberante.

2.3 Problema de localización usando micrófonos

Hay muchas aplicaciones en las que es necesario conocer la posición de una persona u objeto en un entorno interior. Existen diferentes alternativas para esto, algunas de ellas no funcionan en interiores como la localización mediante Global Positioning System (GPS) [30, 31] o por radiobaliza [32], otras son intrusivas (ya que requieren que el usuario lleve encima algún tipo de dispositivo) como la localización mediante sensores Infrarrojos (IR) [26] o ultrasonidos [33, 34], otras pueden no respetar la privacidad de gente como las cámaras [35] que además encarecen el sistema y por último la localización con micrófonos

[36, 37]. El uso de micrófonos permite realizar el posicionamiento en entornos interiores, de forma no intrusiva, y respetando la privacidad de los usuarios. Por este motivo, el uso de micrófonos como receptores para posteriormente calcular la posición del hablante es mucho más aplicable para espacios cerrados que el resto de técnicas.

Un posible caso de aplicación en el que sea requerido calcular el posicionamiento de una o varias personas en un espacio cerrado es para poder crear un mapa de calor del movimiento de la/s persona/s en una sala, una tienda o una parte de un centro comercial para poder usar dicho mapa de calor del movimiento para colocar de manera más precisa los anuncios o los artículos que se quieran vender más.

2.3.1 Introducción a los modelos acústicos

En la actualidad [1], existen tres enfoques diferentes para modelar la acústica de un recinto en función del grado de precisión y complejidad de la determinación del campo sonoro, que se enumeran a continuación:

- Métodos basados en la teoría estadística: son capaces de estudiar el comportamiento de la energía acústica en un recinto de forma rápida y sencilla, obteniendo medidas cualitativas que permiten detectar fallos. Sin embargo para hacer uso de estos métodos es necesario disponer de un campo acústico uniforme en el recinto, con un tiempo de llegada de señales a cada punto equiprobable. Además es necesario que las dimensiones del recinto sean superiores a la longitud de onda y los coeficientes de absorción de las superficies deben ser similares. Dentro de estas técnicas se encuentra el método de análisis de energía estadístico *Static Energy Analysis (SEA)* [38], que predice comportamientos de sonido en sistemas acústicos complejos.

- Métodos basados en la teoría ondulatoria: permiten estudiar la naturaleza ondulatoria del sonido. Para realizarlo es necesario resolver un conjunto de ecuaciones diferenciales parciales complejas. Éstos consideran el aire del interior de un entorno como un sistema vibratorio complejo, formado por varios sistemas de menor complejidad. De forma que en estado estacionario las vibraciones acústicas generan un gran número de ondas estacionarias mientras que cuando la fuente sonora deja de emitir estas ondas se amortiguan de forma exponencial.

  Dentro de estas técnicas se encuentra el método de elementos finitos (*Finite Element Method (FEM)*) [39], que trata de simplificar el problema de solucionar las ecuaciones diferenciales parciales buscando una solución numérica aproximada; el método de elementos de contorno (*Boundary Element Method (BEM)*) [40], que en una primera etapa solo calcula la solución a las ecuaciones en el contorno de la geometría a estudiar, aunque en postprocesamiento deba calcularse en todo el dominio; y el método de diferencias finitas en el dominio del tiempo (*Finite-Different Time-Domain (FDTD)*) [41], que reemplaza las ecuaciones de Maxwell definidas en derivadas parciales por un sistema de ecuaciones en diferencias finitas.

- Métodos basados en la teoría geométrica: están basados en rayos sonoros. El rayo va a determinar la trayectoria de la onda desde que sale de la fuente hasta que llega al objetivo. Estas técnicas calculan la respuesta impulsiva del entorno determinando reflexiones de rayos especulares.

  Dentro de éstas se encuentran el método fuente-imagen (*Image-Source Method (ISM)*) [42–44] que es capaz de determinar con precisión la trayectoria y dirección que han seguido los rayos sonoros desde que salen de la fuente hasta que son recibidos en el objetivo; el método de trazado de rayos (*ray-tracing*) [42, 45] que es una derivación del método de fuente-imagen, evitando buscar todos los trayectos limitando la computación; el método de trazado de haz (*beam tracing*) [46] que deriva del método de trazado de rayos sustituyendo los rayos, que no tienen espesor, por haces, que tienen

espesor. Haciendo que si un haz se cruza con un obstáculo se elimina parte del mismo en vez del haz completo.

Teniendo en cuenta la dificultad del problema que se quiere optimizar, teniendo en cuenta que se tiene que realizar este proceso de forma iterativa para calcular las posibles oclusiones de los objetivos, por simplicidad se va utilizar el método de fuente-imagen que se desarrolla a continuación.

2.3.2 Método de fuente-imagen

Para el posicionamiento en una sala se miden los tiempos de llegada o la diferencia de tiempo de llegada de las señales a los sensores. Esas señales pueden llegar de forma directa o por medio de reflexiones. Esas reflexiones producen unos nuevos caminos para que las señales lleguen a los sensores. La suma de todos los caminos posibles se denomina multicamino.

Al igual que para sensores infrarrojos se ha demostrado que es bastante influyente el efecto del multicamino de la señal recibida, en los micrófonos este efecto es aún mas relevante. Por lo que es necesario tenerlo en cuenta para el posicionamiento.

Para este trabajo debido al alto coste computacional que se requiere al tener que calcular el modelo acústico muchas veces, por simplicidad se ha optado por el método de fuente-imagen o método de las imágenes [47] para calcular las reflexiones especulares con mayor potencia.

En este método las reflexiones del sonido son calculadas formando imágenes especulares de la fuente primaria con respecto a las diferentes superficies reflectantes del entorno. Esas imágenes se consideran como fuentes secundarias que emiten nuevos rayos sonoros con una atenuación relativa al coeficiente de reflexión de la superficie (el cual depende del tipo de material de la superficie), y con un retardo que depende de la distancia entre la fuente y el punto del receptor de la imagen. Dicho análisis asume que las reflexiones en las paredes son especulares, es decir que el ángulo de reflexión es igual al ángulo de incidencia. Además se modelan las ondas sonoras como rayos, entendiéndose por rayo a la línea que indica la dirección y el sentido de propagación del sonido, con un contenido energético que depende de la energía total radiada, del número de rayos emitidos y de la directividad de la fuente.

Las imágenes especulares son los puntos simétricos del destino con respecto a las distintas superficies de reflexión. En la Figura 2.2 se muestra un ejemplo sencillo de la aplicación del método de las imágenes para una habitación rectangular. Siendo S la fuente, R el destino, donde se muestra el camino directo y las reflexiones especulares con cada una de las paredes para un rebote.

Una vez obtenidos los puntos imagen para cada objetivo a analizar, éstos se consideran como nuevos objetivos con una atenuación inversamente proporcional a la distancia al origen y teniendo en cuenta el coeficiente de reflexión de las paredes.

Las principales características/ventajas del método fuente-imagen se indican a continuación:

- Funcionamiento: este método destaca por su sencillez y por la obtención de soluciones exactas para recintos con superficies bien definidas. Se trata de un método recursivo y automatizable con estos pasos:

  1. El primer paso consiste en reflejar la fuente (S) en todas las superficies del recinto, obteniendo así un conjunto de fuentes imagen.
  2. El siguiente paso, e iterativo, consiste en seleccionar cada una de las fuentes imagen obtenidas y volver a reflejar cada una de ellas con las superficies que la rodean para obtener los siguientes rebotes. Debe existir una condición de terminación para que el método concluya, ya sea por el número de reflexiones o bien por la longitud máxima que recorren los rayos.

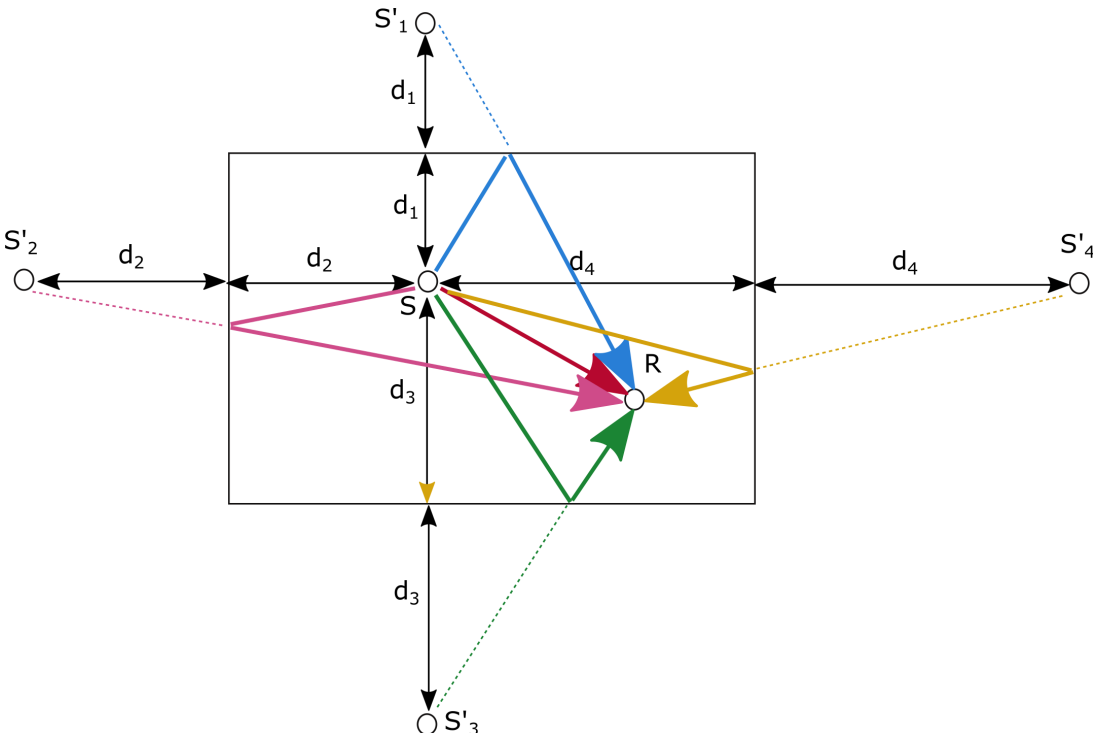


Figura 2.2: Ejemplo de aplicación del método de las imágenes para una habitación rectangular [1]

La posición de las fuentes imágenes (S'), se calcula teniendo en cuenta que tiene que estar a la misma distancia de la superficie reflectante que la fuente original, como puede verse en la Figura 2.3.

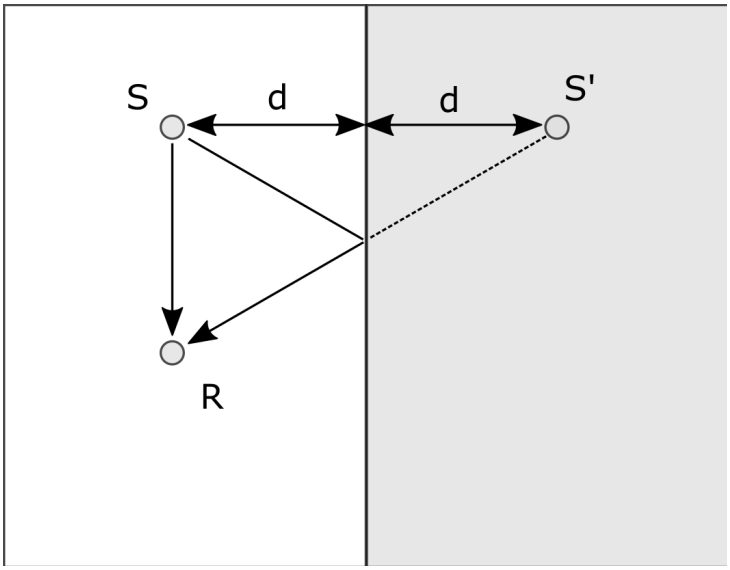


Figura 2.3: Ejemplo de cálculo de fuente imagen [1]

- Comprobaciones: el procedimiento del método fuente imagen es sencillo en recintos con forma de cuboide y sin obstáculos, pero para formas arbitrarias y/o con obstáculos existe el problema que algunas fuentes imagen pueden no contribuir al campo sonoro total, por lo que se deben realizar las siguientes pruebas:

  1. Validez: hay que comprobar que la fuente imagen fue creada por una superficie reflectante, en caso contrario, no será valida.

2. Proximidad: se establecen o bien el número de reflexiones máximas o la distancia recorrida máxima y en caso de que no cumplan uno de esos criterios las fuentes imagen deben ser descartadas.

3. Visibilidad: hay que tratar de averiguar si la fuente imagen creada es visible en el objetivo. Para las fuentes de primer orden, la fuente imagen es visible si la recta trazada entre ésta y el oyente pasa por el plano de la superficie reflectante, en un punto dentro de los límites de la superficie, por lo que es necesario obtener el punto de corte con esa superficie. Para un número de orden mayor el proceso puede hacerse de forma iterativa, ya que el punto de corte calculado previamente se comporta como una nueva fuente y usando ese punto y la siguiente fuente imagen puede realizarse el mismo proceso que para las fuentes imagen de primer orden.

2.4 Definición del modelo basado en SRP

Este apartado ha sido adaptado de [16].

La localización en el contexto de agrupaciones de micrófonos, ya sean colocados de forma aislada o en arrays, puede llevarse a cabo mediante algoritmos basados en TDOA explicado en apartados anteriores o empleando técnicas basadas en SRP. A diferencia de los basados en TDOA que necesitan un único sensor de referencia, los basados en SRP calculan la diferencia de tiempos en función de pares, por lo que son menos sensibles a los obstáculos.

Los algoritmos basados en SRP se basan en la capacidad de los micrófonos de apuntar a una determinada dirección o posición del espacio, hacen uso de técnicas de *beamforming* [48]. En el uso de esta técnica para localización, a la salida del *beamformer* se le conoce como *steered response*.

Una vez obtenidos los puntos de fuente imagen especular y teniendo en cuenta la absorción acústica de las paredes se procede a obtener el modelo SRP-PHAT para calcular el mapa de potencia acústica asociado a cada punto del espacio.

Por simplicidad, se considera el escenario con propagación anecoica de [17] donde una única fuente acústica localizada en la posición $\mathbf{r}$ genera una señal en banda base con un ancho de banda ω_0. En ese caso, la señal en el micrófono i puede representarse como un versión atenuada y desplazada en el tiempo de $X(\omega)$, p.e. $X_i(\omega) = \alpha_i\, X(\omega) \mathrm{e}^{-\mathrm{j}\omega\tau_i(\mathbf{r})}$, donde α_i es un factor de atenuación dependiente del micrófono, y $\tau_i(\mathbf{r})$ es el retardo de propagación entre $\mathbf{m}_i$ y $\mathbf{r}$, que es calculada como $\tau_i(\mathbf{r}) = \frac{1}{c}\|\mathbf{r} - \mathbf{m}_i\|$.

En ese caso, la ecuación (2.11) puede simplificarse como:

$$\begin{aligned} P(\mathbf{q},\mathbf{r}) &= \sum_{i=1}^{N_\mu}\sum_{j=1}^{N_\mu} \int_{-\omega_0}^{\omega_0} 2\cos\left(\omega\left(\Delta\tau_{ij}(\mathbf{q}) - \Delta\tau_{ij}(\mathbf{r})\right)\right) \mathrm{d}\omega = \\ &= \sum_{i=1}^{N_\mu}\sum_{j=1}^{N_\mu} \left[\frac{4\sin\left(\omega_0\left(\Delta\tau_{ij}(\mathbf{q}) - \Delta\tau_{ij}(\mathbf{r})\right)\right)}{\Delta\tau_{ij}(\mathbf{q}) - \Delta\tau_{ij}(\mathbf{r})}\right], \end{aligned} \tag{2.12}$$

donde se tiene una dependencia explícita incluida con $\mathbf{r}$, y $\Delta\tau_{ij}(\mathbf{q}) = (\tau_i(\mathbf{q}) - \tau_j(\mathbf{q}))$ es la diferencia en tiempo de llegada de la señal acústica en los micrófonos $\mathbf{m}_i$ y $\mathbf{m}_j$ desde $\mathbf{q}$. También puede verse como el retardo en apuntarse que tiene el par de micrófonos $(\mathbf{m}_i, \mathbf{m}_j)$ a la posición $\mathbf{q}$. Cada uno de los términos en la suma de la ecuación (2.12) corresponde al modelo Generalized cross-correlations PHAT (GCC-PHAT) para cada par de micrófonos.

2.5 Obtención de la *Spatial Likelihood Function (SLF)*

Este apartado ha sido adaptado de [16].

En este trabajo, y por simplicidad, las posiciones exploradas $\mathbf{q}_i$ y las posibles posiciones de la fuente $\mathbf{r}_j$ van a ser evaluadas en la misma rejilla regular, en el espacio tridimensional cubriendo todo el entorno (con $i, j = 1 \dots Q$).

Entonces, dado mapa de potencia de SRP-PHAT obtenido a partir de la ecuación (2.12) para cada posición geométrica $\mathbf{q}_i$ en el área explorada para cada fuente localizada en $\mathbf{r}_j$, asumimos que la SLF para cada una de esas posiciones es proporcional a dicho mapa de potencia[1], como:

$$f_{\mathcal{M}}(\mathbf{q}_i|\mathbf{r}_j) = \alpha_j \cdot (P(\mathbf{q}_i, \mathbf{r}_j) + \beta_j) \tag{2.13}$$

El cálculo de α_j y β_j se realiza imponiendo a la SLF comportarse como tal, es decir, haciendo que sea una función todo positiva, y que la suma de todos sus valores sea igual a uno, para cada $\mathbf{r}_j$:

$$\begin{aligned} f_{\mathcal{M}}(\mathbf{q}_i|\mathbf{r}_j) &\geq 0, \quad \forall \mathbf{q}_i, \mathbf{r}_j \\ \sum_{i=1}^{Q} f_{\mathcal{M}}(\mathbf{q}_i|\mathbf{r}_j) &= 1, \quad \forall \mathbf{r}_j \end{aligned} \tag{2.14}$$

Por tanto, finalmente la SLF se calcula como(con $i, j, k, l = 1 \dots Q$):

$$f_{\mathcal{M}}(\mathbf{q}_i|\mathbf{r}_j) = \frac{P(\mathbf{q}_i, \mathbf{r}_j) - \min_{\forall k}(P(\mathbf{q}_k, \mathbf{r}_j))}{\sum_{\forall l}\left[P(\mathbf{q}_l, \mathbf{r}_j) - \min_{\forall k}(P(\mathbf{q}_k, \mathbf{r}_j))\right]}, \quad \forall \mathbf{q}_i|\mathbf{r}_j \tag{2.15}$$

En la Figura 2.4.b se muestra un ejemplo de la SLF estimada para todos los $\mathbf{q}_i$ en el espacio y para una fuente acústica localizada en $\mathbf{r}_j = (0, -3, 0)^\top$ para una sala rectangular (cuya planta se muestra en la Figura 2.4.a) con 3 arrays de micrófonos (con un total de 12 micrófonos). La distribución de los valores de $f_{\mathcal{M}}(\mathbf{q}_i|\mathbf{r}_j)$ en el entorno depende de $\mathcal{M}$. La SLF de la Figura representa la probabilidad con la que el objetivo puede encontrarse en cada punto del espacio a una determinada altura (calculada con las coordenadas tridimensionales de cada punto de la rejilla).

Como se puede observar se aprecian formas hiperbólicas que se asemejan al proceso de triangulación hiperbólica [24], por lo que se asume que esta estimación es válida.

2.6 Error de localización

2.6.1 Cálculo del error de localización

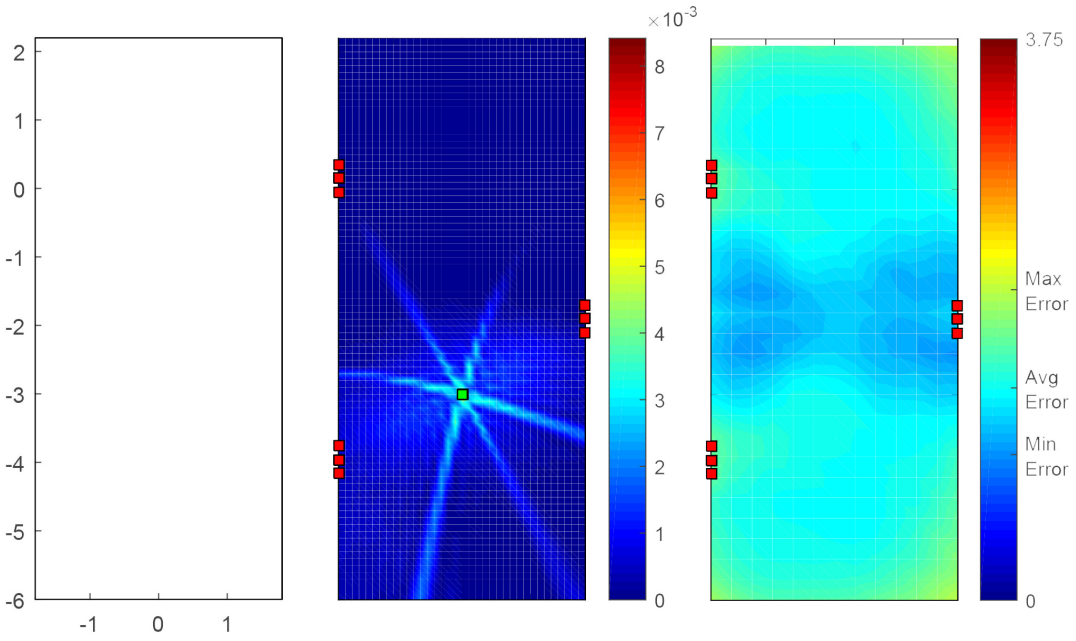
Este apartado ha sido adaptado de [16].

Usando la SLF definida en la Ecuación (2.15), se puede obtener el error medio de localización (representado como $\varepsilon_{\mathcal{M}}(\mathbf{r}_j)$) para cada posición explorada en el entorno, considerando el error medio como la contribución de cada posible localización de la fuente posible $\mathbf{r}_j$:

$$\varepsilon_{\mathcal{M}}(\mathbf{r}_j) = \sum_{i=1}^{Q} \|\mathbf{q}_i - \mathbf{r}_j\| \cdot f_{\mathcal{M}}(\mathbf{q}_i|\mathbf{r}_j), \quad \forall \mathbf{r}_j \tag{2.16}$$

[1] Esta asunción parece razonable, ya que las posiciones con mayor potencia acústica deberían ser más propensas a corresponder, efectivamente, a una fuente acústica activa. Sin embargo, se llevarán a cabo experimentos adicionales en el futuro, para apoyar formalmente esta suposición.

La Figura 2.4.c muestra el error medio de localización estimado para la misma configuración que la Figura 2.4.b.



(a) Planta de una sala

(b) SLF, para un punto concreto a una altura determinada

(c) Error para cada punto

Figura 2.4: Ejemplo de planta de una sala, SLF y función de error.

2.6.2 Uso de arrays de micrófonos

Este apartado ha sido adaptado de [15].

En los apartados anteriores se ha hablado de arrays de micrófonos, en este trabajo se consideran tanto micrófonos individuales como arrays de 4 micrófonos colocados en forma de T invertida con una separación horizontal de 20cm y una separación vertical de 30cm, como las mostrasdas en las Figuras 2.5 y 2.6. La elección de esta distribución geométrica se debe a que fue la usada en el proyecto CHIL [49].

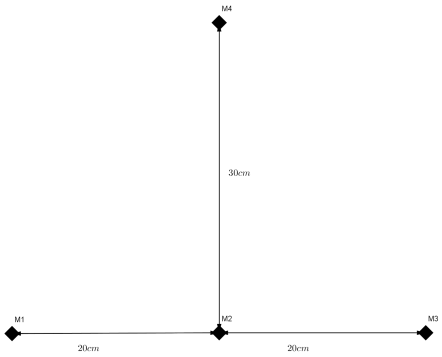


Figura 2.5: Esquema array de micrófonos utilizado en este trabajo

Esta limitación ha de tenerse en cuenta en la etapa de posicionamiento de los micrófonos. Teniendo en cuenta que que la posición de cada unos de los micrófonos es dependiente de la del resto, en la etapa de posicionamiento se buscará únicamente la posición de uno de ellos (en este caso el central), para calcular posteriormente las posiciones del resto.

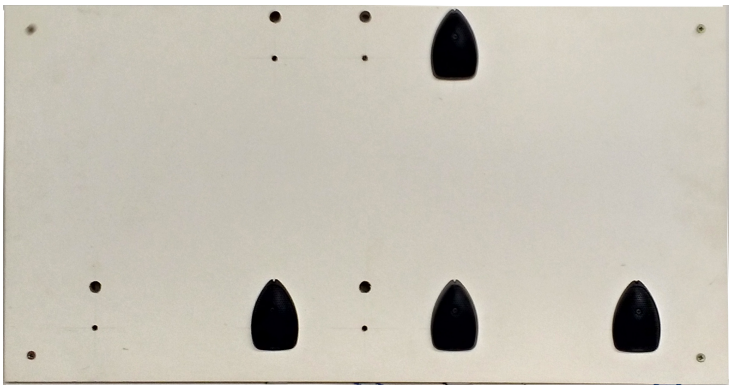


Figura 2.6: Array real de micrófonos utilizado en este trabajo

Al igual que para sensores de otro tipo para el cálculo de la posición dentro del espacio en tres dimensiones es necesario el uso de cuatro micrófonos, por lo que en el caso de este tipo de arrays sería posible el posicionamiento con un único array.

2.7 Explicación teórica del problema de posicionamiento de sensores

2.7.1 Definición del problema

Este apartado ha sido adaptado de [15].

La ubicación de los sensores (y en especial de micrófonos) cambia considerablemente el error de posicionamiento obtenido, por eso es importante tratar de buscar la posición óptima de los sensores. El problema de éste cálculo es la infinidad de soluciones posibles. En el siguiente ejemplo se puede apreciar cómo, al aumentar el número de sensores, las posiciones de éstos se incrementan exponencialmente, considerando posiciones como ubicaciones posibles del conjunto de sensores y no de cada sensor en particular:

Suponiendo que se limitan las posiciones de cada sensor a 100 posibles ubicaciones en una habitación, a continuación se analiza cómo aumentan las combinaciones posibles de posiciones de los sensores con el número de estos:

- 1 sensor: 100 distribuciones posibles.
- 2 sensores: $\binom{100}{2} = 4950$ distribuciones posibles.
- 3 sensores: $\binom{100}{3} = 161700$ distribuciones posibles.

  .

  .

  .

- N sensores: $\binom{100}{N}$ distribuciones posibles.

Aunque 100 posiciones es un número muy bajo para un ejemplo real en el que las posiciones posibles superan con facilidad los millones (p.e. para una habitación cuadrada de 3·3·3m con una rejilla de 1cm entre cada posición y con la posibilidad de ubicar los sensores en las paredes y el techo se tienen 540.000

posiciones posibles, esto permite ver cómo calcular todos los resultados y elegir el mejor podría suponer un tiempo no asumible, ya que podría superar los años en un caso real.

Para reducir notablemente el tiempo de estimación de la ubicación óptima es recomendable el uso de algoritmos metaheurísticos. De forma general este tipo de algoritmos permiten obtener una solución más o menos satisfactoria (dependiendo del algoritmo en concreto utilizado y del número de iteraciones) en un tiempo razonable. Los algoritmos evolutivos, y en concreto el algoritmo genético, son unos de los métodos más utilizados para el posicionamiento óptimo de sensores (p.e. para sensores IR [26], para sensores de ultrasonidos [50], para balizas [51–53] y redes de sensores inalambricos [54]) y micrófonos (p.e. para el posicionamiento de micrófonos y altavoces[55], para la ubicación de arrays de micrófonos utilizando técnicas híbridas [56], para controlar el ruido dentro de un avión [57], para controlar la radiación sonora de campo lejano con micrófonos y actuadores piezoeléctricos [58] y para mejorar la adquisición de la voz en el interior de un vehículo [59]).

2.7.2 Descripción del algoritmo genético mono-objetivo

Un algoritmo genético [14] es un algoritmo metaheurístico que imita la teoría de la evolución de Charles Darwin para la resolución de problemas.

Para ello se parte de una población inicial que cumple los requisitos del problema, creada de forma aleatoria, de la cual se seleccionan los individuos más capacitados para luego reproducirlos o cruzarlos y mutarlos de forma iterativa con el objetivo de obtener una siguiente generación de individuos mejor adaptados que los de la anterior generación.

En la Figura 2.7 se puede ver el diagrama de flujo de un algoritmo genético genérico:

Esta estructura general se puede adaptar fácilmente a cada problema de la siguiente manera:

1. Se define de la mejor forma posible la población.
2. Se crea la población inicial de forma aleatoria cumpliendo los requisitos del problema.
3. Se definen las funciones de cruce y mutación.
4. Se crea la función de evaluación de la población o función objetivo. Dicha función se centra en evaluar cada individuo de la población asignándole un valor para que después la función de selección pueda escoger los mejores individuos para trabajar con ellos.

   La función objetivo puede adaptarse al problema para garantizar que se cumplan ciertas características de forma que si no las cumplen se penalice esa solución.

2.7.2.1 Codificación del problema

Con el fin de definir la población de la mejor forma posible, se puede optar por distintos tipos de codificaciones:

- Binaria: creando un vector de tamaño Y de unos y ceros donde tiene que haber X unos que representan las estaciones base seleccionadas y los ceros las que no (o viceversa), donde las cada posición del vector representa una posición del espacio fijada previamente.
- Decimal: creando un vector de tamaño X con elementos entre los Y posibles.
- Real: guardando la posición (x,y) de los X elementos en un vector de tamaño 2X, o de 3X si las posiciones son de tres dimensiones.

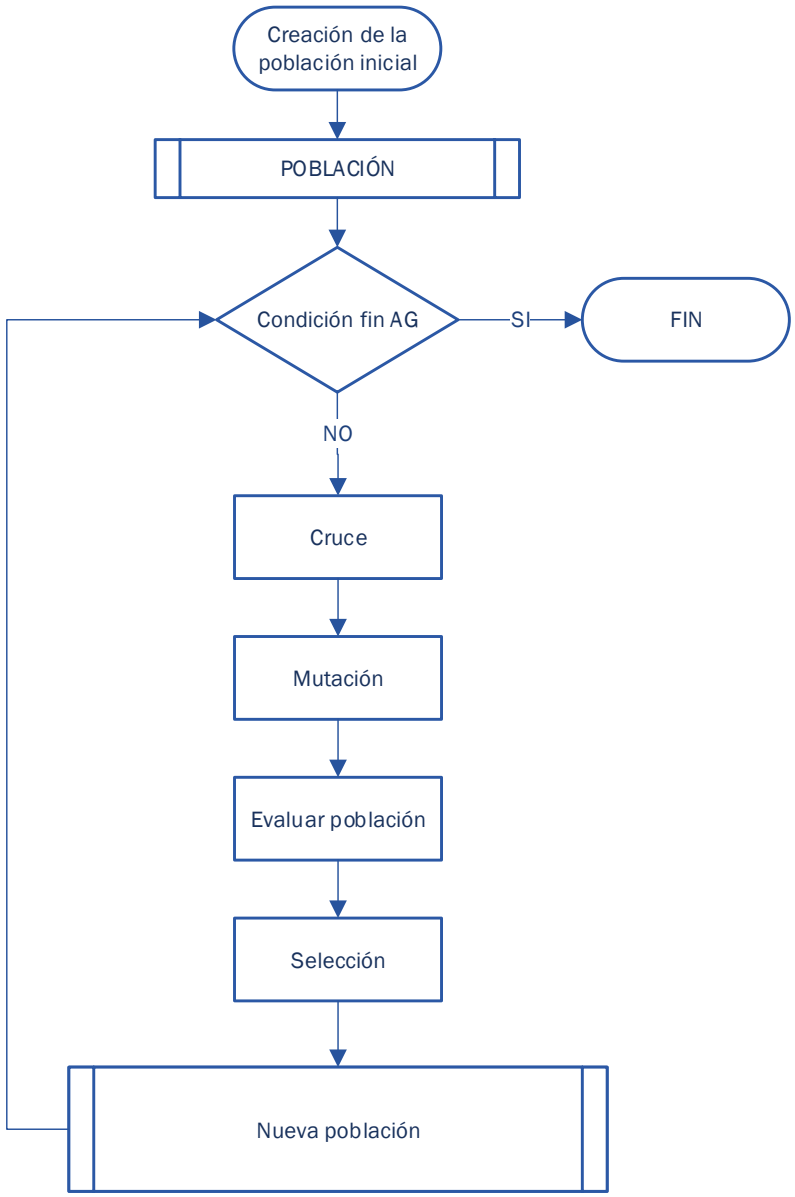


Figura 2.7: Diagrama de bloques general de un algoritmo genético

La codificación real sólo es práctica para un caso *Greedy* en el que pueden posicionarse en cualquier punto del espacio, y donde el número de posiciones posibles es muy grande. Las codificaciones binaria y decimal son más prácticas para un caso *Brownfield*, o donde el número de posiciones posibles es menor y el tamaño de las matrices generadas son de menor tamaño.

2.7.2.2 Creación de la población inicial

Una vez definido el tipo de codificación, ha de crearse la población inicial (de tamaño s_p) de forma aleatoria, asegurándose de que todos los elementos de la población inicial cumplen los requisitos del problema.

2.7.2.3 Procedimiento de Selección

Una vez generada la nueva población es necesaria una selección en base al *ranking* de resultados de la función de *fitness* (o función objetivo) para que el tamaño de la población vuelva a ser como la inicial, eliminando los elementos repetidos (para no crear superindividuos) y los que tengan peor resultado de función objetivo.

Como se ha comentado previamente, la selección puede hacerse en base a distintos criterios, y en distinto orden. Los criterios de selección pueden ser los siguientes entre otros:

- *Ranking*: se seleccionan los mejores después del cruce y mutación, junto con la población anterior.
- Elitismo: se seleccionan los mejores para pasarlos directamente a la siguiente población, sin tener que volver a evaluarlos.
- Torneo probabilístico: Las nuevas soluciones compiten por quedarse, pudiendo darse casos que con cierta probabilidad soluciones "malas" solapen soluciones "buenas".

Pueden usarse unos u otros en función del problema, siendo los más comunes el de *Ranking* y Elitismo.

En base al criterio que se haya elegido la selección puede hacerse en momentos distintos del algoritmo, p.e. si se establece el uso de *ranking* el proceso de selección debe hacerse después del cruce y mutación para que el tamaño de la población sea el mismo, mientras que si se aplica elitismo el proceso de selección puede hacerse o bien al inicio como en el diagrama general, o bien al final si se establece que la suma entre las soluciones elitistas y el número de cruces o mutaciones puede ser mayor que el de la población inicial.

Además, la selección puede hacerse teniendo en cuenta que la nueva población no tenga elementos repetidos o no, siendo lo más efectivo para evitar superindividuos que no se repitan.

Si el tiempo de evaluación es muy elevado pueden usarse además los criterios de *ranking* y elitismo (guardando toda la población anterior y su evaluación) para no tener que evaluar de nuevo la población anterior y sólo tener que evaluar los nuevos elementos generados mediante los procesos de mutación y cruce.

2.7.2.4 Procedimiento de Cruce

El procedimiento de Cruce, o reproducción sexual, sirve para crear nuevos individuos a partir de los elementos de la población anterior, con características de estos, con la intención de que los nuevos hijos mejoren el objetivo que sus respectivos predecesores. Al igual que la selección el proceso de cruce puede hacerse en base a distintos criterios y en distinto momento. El momento depende únicamente de la selección, pero el criterio puede ser de distintos tipos independientemente del criterio de selección elegido. Pudiendo elegirse entre los siguientes entre otros:

- Cruce por un punto: se realiza el cruce de forma que fijando un punto de cruce aleatoriamente dentro del vector de solución de los padres y seleccionando las componentes de un padre a un lado del punto de cruce y del otro al otro lado y la combinación contraria, se obtienen dos hijos con información de ambos padres como en la Figura 2.8.
- Cruce por dos puntos: de forma análoga al cruce por un punto pero seleccionando aleatoriamente 2 puntos de cruce se realiza el cruce como en la Figura 2.9.

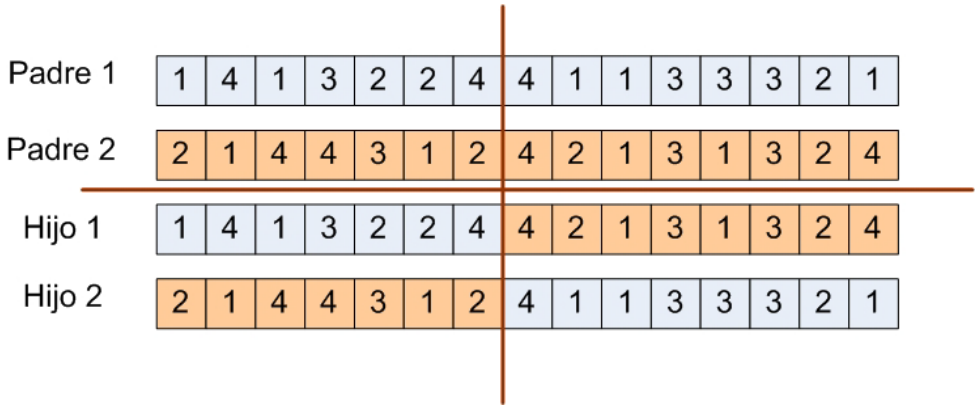


Figura 2.8: Cruce por un punto

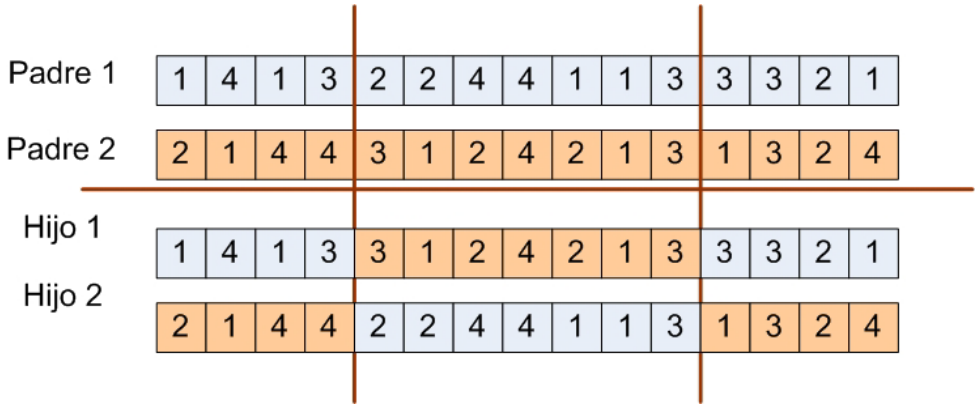


Figura 2.9: Cruce por dos puntos

- Cruce por punto intermedio: se calcula una posición intermedia de los padres, $x_{x_i}^{(l)} = x_{i_1}^{(l)} + \frac{1}{2} \cdot (x_{i_1}^{(l)} - x_{i_2}^{(l)})$, donde $x_{x_i}^{(l)}$ es el $i$esimo hijo de la $l$esima generación y $x_{i_k}^{(l)}$ es el $k$esimo padre del $i$esimo hijo.

La selección de estos viene determinada por el tipo del problema, ya que no todas tienen sentido para todas las posibilidades, p.e. el cruce por punto intermedio sólo tiene cabida cuando tenemos un problema de infinitas posibilidades o en un espacio real, donde tiene sentido calcular una posición intermedia entre las soluciones anteriores.

Además, puede aplicarse corrección cuando el resultado del cruce no tiene sentido, cuando dentro de una misma población existen elementos iguales o cuando está fuera de los límites de posicionamiento. Esto puede hacerse de dos formas, corrigiendo los elementos no válidos a una posición cercana que si es válida o sustituyéndolo por un elemento aleatorio que si cumple las especificaciones.

2.7.2.5 Procedimiento de Mutación

El procedimiento de Mutación sirve para modificar un porcentaje de soluciones de forma aleatoria con el fin de evitar caer en máximos o mínimos locales. Al igual que los procedimientos anteriores puede hacerse en base a distintos criterios y en distinto momento. En función del momento podemos establecer varias alternativas:

- Si se considera que las soluciones también pueden crearse por reproducción asexual, podría tomarse la mutación como si de este tipo de reproducción se tratara para generar nuevos individuos a partir de la población inicial mutando parte del cromosoma de la solución.

- Partiendo de la reproducción sexual creada anteriormente en el proceso de cruce puede, con un porcentaje dado, modificarse parte de los mismos para mejorar la solución.

Además, puede hacerse en base a varios criterios seleccionando el número de cromosomas que quieren mutarse, estableciendo el número de puntos de la mutación.

Siendo $x_{m_i}^{(l)} = x_i^{(l)} + r_{m_i}$ los nuevos elementos mutados, donde r_{m_i} es una aleatorio Gaussiano del mismo tamaño que x_{m_i}, con valores distintos de cero en las posiciones que quieren mutarse.

En la Figura 2.10 puede verse de forma gráfica este proceso.

Figura 2.10: Mutación por 1 o más puntos

2.7.2.6 Procedimiento de Evaluación

El procedimiento de evaluación es el encargado de calcular la función de *fitness* para cada individuo de la población con el fin de poder saber cuáles de esos individuos obtienen mejor resultado de la función objetivo.

2.7.2.7 Condición de fin del Algoritmo Genético

La condición de fin del algoritmo genético puede hacerse en base a dos criterios, o bien si se alcanzar el número de generaciones deseado o bien buscando el punto donde se alcanza la convergencia. Dicha convergencia puede calcularse de varias formas, en base a una tolerancia (fijando la tolerancia mínima de variación entre experimentos) o fijando un intervalo de iteraciones máximo tras el cual si la solución optima no ha cambiado se determina que se ha alcanzado la convergencia.

2.7.3 Descripción teórica del algoritmo genético multi-objetivo

Como se hizo patente en el trabajo [16] en ocasiones es necesario optimizar la solución no sólo teniendo en cuenta un objetivo sino varios.

Por esta razón aparecen los algoritmos metaheurísticos multi-objetivo como el *Vector Evaluated Genetic Algorithm (VEGA)* [60, 61], *Variable Objective Weighting Genetic Algorithm (VOW-GA)* [62],

Random Weights Genetic Algorithm (RW-GA) [63], *Nondominated Sorting Genetic Algorithm (NSGA)* [64], *Nondominated Sorting Genetic Algorithm II (NSGA-II)* [3], *Pareto Archived Evolution Strategy (PAES)* [65], *Strength Pareto Evolutionary Algorithm (SPEA)* [66] y *Strength Pareto Evolutionary Algorithm 2 (SPEA)* [67].

La definición de como se tratan los distintos objetivos en los distintos tipos de algoritmos multi-objetivo es fundamental, donde aparecen dos grandes grupos de procesos, utilizando pesos o basados en soluciones no-dominadas.

La principal diferencia entre los algoritmos citados es el criterio de selección del algoritmo genético, ya sea en la forma de ajustar los pesos, o bien en la técnica de calcular los paretos de soluciones.

2.7.3.1 Algoritmos genéticos multi-objetivo utilizando pesos

Este grupo incluye los algoritmos que no incorporan el concepto de óptimo de Pareto en el mecanismo de selección del algoritmo evolutivo sino que usan funciones agregativas lineales. Estos implementan un modelo evolutivo utilizando pesos para la agregación de los objetivos. Entre ellos están VEGA, VOW-GA, RW-GA, etc.

Estos algoritmos proponen mecanismos de asignación de objetivos relativamente sencillos que no garantizan una correcta resolución del problema.

El mecanismo de asignación de objetivos puede realizarse por combinación lineal de objetivos o calculando el objetivo como una suma ponderada, donde los pesos se fijan a priori.

Las ventajas de usar estos algoritmos frente a otros es que son fáciles de entender y formular, ya que a fin de cuentas adecuan un problema multi-objetivo a un problema mono-objetivo, sin embargo esa priorización o sistema de pesos puede ser arbitrario debido a una sobresimplificación del problema o una pobre comprensión del mismo por lo que no es válido para algunos problemas multi-objetivo, sobretodo en los problemas donde los distintos objetivos pueden tener cierta dependencia unos con otros.

2.7.3.2 Algoritmos genéticos multi-objetivo basados en soluciones no dominadas

Estos algoritmos multi-objetivo se rigen por la misma estructura que los algoritmos mono-objetivo excepto el proceso de selección, donde incorporan el concepto de soluciones pareto-optimales o no-dominadas. Entre ellos están NSGA, NSGA-II, PAES, SPEA, etc.

Se dice que una solución a domina a otra b y se escribe como $a \prec= b$ si es mejor en todos los objetivos, o igual en todos los objetivos y al menos mejor en uno de ellos. Las soluciones que no son dominadas por otra se llaman Pareto-optimales o no dominadas.

No suele existir una solución óptima única para un problema multi-objetivo, si no que existe un conjunto de soluciones no-dominadas que forman el Frente de Pareto.

En la Figura 2.11 se muestran los distintos frentes de pareto que se pueden obtener para un mismo espacio de soluciones con dos objetivos dependiendo de la optimización para cada objetivo deseada.

El fin principal de este tipo de algoritmos es muestrear adecuadamente dicho frente de pareto para obtener el mayor número de soluciones óptimas posibles, permitiendo realizar la toma de decisiones a posteriori.

Para obtener dicho frente de pareto este trabajo se centra en la técnica utilizada en el algoritmo NSGA-II desarrollado en [3], ya que genera buen resultado en comparación al resto de algoritmos citados previamente. La selección de este algoritmo sigue el diagrama de la Figura 2.12, donde el proceso de

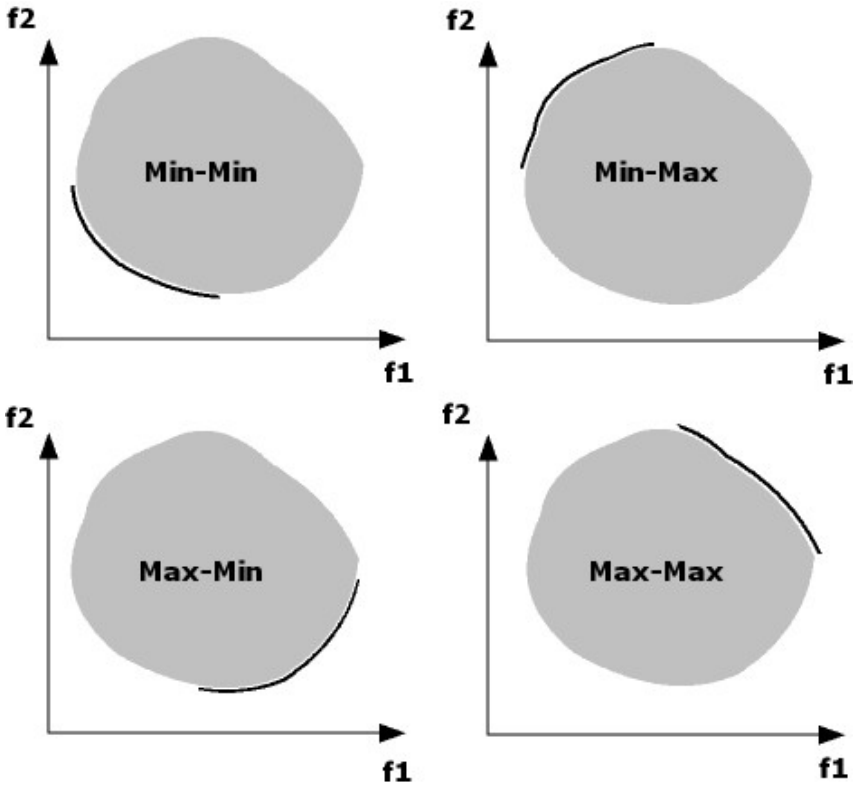


Figura 2.11: Diferentes frentes de pareto para un mismo espacio de soluciones [2]

Non-dominated sorting se encarga de ordenar la población en distintos frentes de pareto no dominados y el proceso *Crowding distance sorting* se encarga de ordenar las distintas soluciones de cada pareto en función de la distancia *Crowding* (en función de la concentración de soluciones alrededor de cada solución), para finalmente descartar las peores soluciones en base a esta ordenación

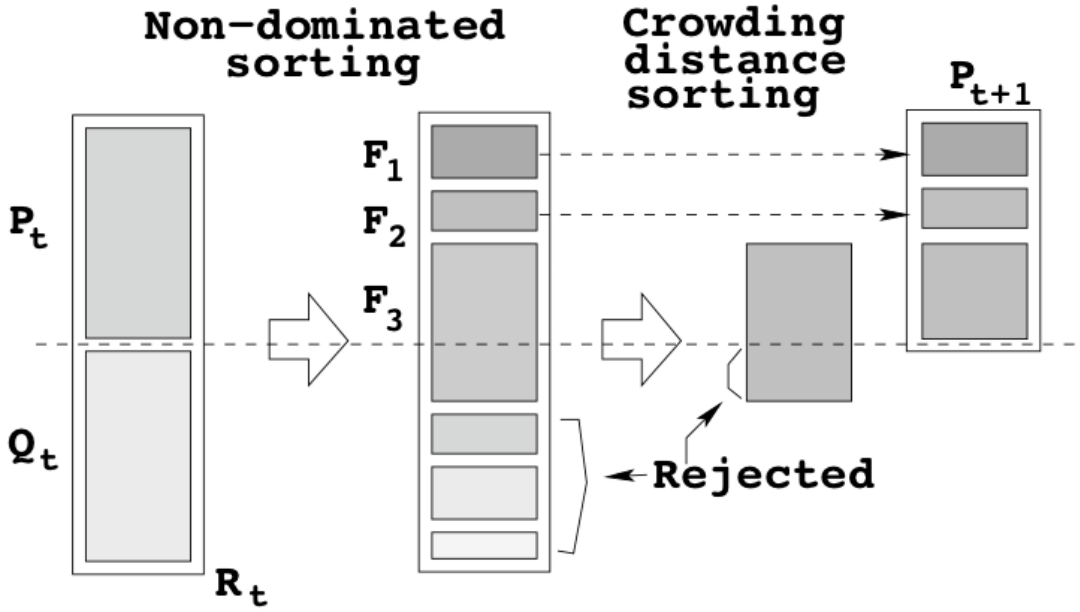


Figura 2.12: Diagrama esquemático del proceso de selección de NGSA-II [3]

Como en el caso de este trabajo se quieren minimizar ambos objetivos, en la Figura 2.13 se puede ver un ejemplo de solución de un algoritmo multi-objetivo, donde los cuadrados representan posibles soluciones:

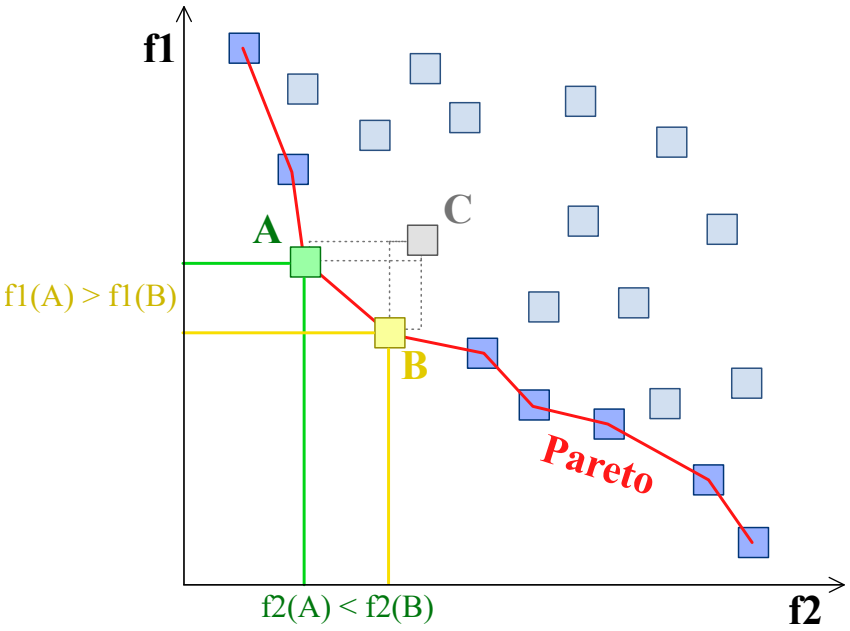


Figura 2.13: Ejemplo de frente de pareto de soluciones [4]

Como se puede ver en vez de una solución única el algoritmo multi-objetivo proporciona un conjunto de soluciones que optimizan en conjunto ambos objetivos. Ese conjunto de soluciones forman el frente del pareto de soluciones que optimizan los objetivos dentro del espacio de decisión del algoritmo genético.

Capítulo 3

Desarrollo

Si la hipótesis sobrevive a la prueba experimental, aumenta su prestigio, y al ir siendo aceptada se desarrolla y se extiende en una forma cada vez más comprensiva.

Max Planck

3.1 Introducción

En este capítulo se explica con detalle el sistema desarrollado en este trabajo. Explicando su funcionamiento interno, los conjuntos de clases definidos, así como los datos de entrada y configuración.

A continuación, se describe el método propuesto en este trabajo para calcular el error de localización basado en las técnicas de localización TDOA y SRP explicadas en el Capítulo 2.

Por último se define el método propuesto para optimizar dicho error tanto utilizando uno o varios objetivos, utilizando algoritmos genéticos mono-objetivo y multi-objetivo.

3.2 Sistema desarrollado

Este trabajo parte del sistema de posicionamiento óptimo de sensores y de arrays de micrófonos para la localización de personas u objetos desarrollado en [15] y [16], para cualquier tipo de sala con algoritmos genéticos mono-objetivo o multi-objetivo.

El sistema desarrollado en este trabajo extiende la técnica de posicionamiento óptimo de sensores de Francisco Domingo Pérez [26] para habitaciones de cualquier forma, para localizar a personas u objetos en el espacio tridimensional y pudiendo poner restricciones de ubicación a los sensores y obstáculos.

Se usan además arrays de micrófonos con una topología específica, formados por cuatro micrófonos colocados en forma de T invertida definidos por la separación vertical y horizontal de los micrófonos, como el mostrado en la Figura 2.5, utilizando una técnica de localización basada en SRP-PHAT. Además, también permite el uso de micrófonos aislados, así como sensores que usen técnicas de localización basadas en TDOA.

3.2.1 Objetivo y descripción general del sistema desarrollado

El objetivo del sistema desarrollado es obtener la ubicación óptima de los sensores o arrays de micrófonos en una sala con unas características concretas.

El sistema permite además escoger cómo se tienen en cuenta los obstáculos que haya en la sala, y aunque pueda mostrar todos los obstáculos, sólo se permite tener en cuenta a la hora de calcular el error en la medida los obstáculos fijos (columnas).

Además, el sistema está diseñado para forzar la ubicación de los sensores en las paredes, el suelo, el techo o combinaciones de éstos.

Todo esto puede configurarse de manera sencilla a través de una interfaz, que ayuda a seleccionar el tipo de sensores a utilizar para el posicionamiento, los obstáculos a tener en cuenta, las restricciones de paredes, altura mínima para colocar los sensores en las paredes, configurar los parámetros del algoritmo genético,...

3.2.2 Datos de entrada y configuración

Aunque la interfaz permite seleccionar la sala para la cual se quiere realizar el proceso de posicionamiento de los sensores, para configurar las características de dicha sala es necesario definirla previamente en modo texto en un archivo "*.env*" como el de la Figura 3.1, que a su vez llama a un conjunto de superficies definidas en ficheros "*.srf*" como el de la Figura 3.2

```
[EnvironmentInfo]

content = Environment definition file
version = 1.1
comment = IDIAP Demo room (for av16.3 simulation)

numSurfaces = 7

[Environment]

dirEnvironments = environments/IdiapRoom/

[SurfaceFiles]

surface0 = leftWall.srf
surface1 = rightWall.srf
surface2 = backWall.srf
surface3 = frontWall.srf
surface4 = floor.srf
surface5 = ceiling.srf
surface6 = table.srf
```

Figura 3.1: Ejemplo de archivo ".env" de definición de sala.

El archivo "*.env*" estará dividido en 3 partes.

1. Una primera parte encabezada por [*EnvironmentInfo*] contiene la información sobre las características generales de la sala:
   - content: Contenido del fichero
   - version: Número de versión.

```
[SurfaceInfo]

content = Surface definition file
version = 1.1
comment = Idiap demo room

numVertexes = 4

[Vertexes]

vertex0 = 1800 2200 1670
vertex1 = 1800 −6000 1670
vertex2 = −1800 −6000 1670
vertex3 = −1800 2200 1670
```

Figura 3.2: Ejemplo de archivo ".srf" de definición de superficie.

- comment: Nombre de la sala.
- numSurfaces: Número de superficies que forman la sala.

2. Un segundo apartado encabezado por $[Environment]$ contiene la ruta relativa de los ficheros de la sala.

3. El tercer apartado encabezado por $[SurfaceFiles]$ que contiene los nombres de los ficheros "$.srf''$ de las superficies que forman la sala, de la siguiente forma: $surfaceN = nombre.srf$. El nombre del fichero además debe contener información relevante para saber de que tipo de superficie se trata, si es una pared (debe contener *wall*), si es el techo (debe contener *ceiling*), si es el suelo (debe contener $floor$), si es una columna o un obstáculo fijo (debe contener *column*), el resto del nombre del fichero y el resto de superficies es independiente, aunque es recomendable denominarlo de forma coherente.

A su vez los archivos "$.srf''$ estarán dividido en 2 partes.

1. Una primera parte encabezada por $[SurfaceInfo]$ contiene la información sobre las características generales de la sala:

   - content: Contenido del fichero
   - version: Número de versión.
   - comment: Nombre de la sala.
   - numVertexes: Número de vértices que forman la sala.

2. El segundo apartado encabezado por $[Vertexes]$ contiene las coordenadas [x y z] de cada vértice de la superficie., de la siguiente forma: $vertexN = xyz$

3.2.3 Interfaz gráfica

Con el fin de facilitar la selección de parámetros del algoritmo y la visualización de los resultados se ha creado la Interfaz Gráfica de Usuario (GUI) mostrada en la Figura 3.3 cuyo funcionamiento se detalla en el Apéndice B.

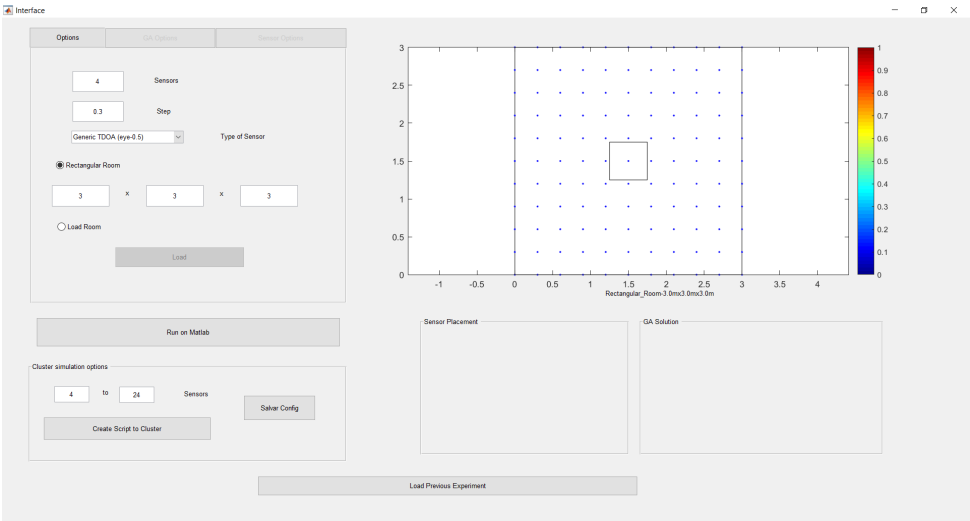


Figura 3.3: Interfaz gráfica desarrollada.

3.2.4 Definición de clases y funciones

En este trabajo se han desarrollado y adaptado 11 funciones, así como 3 clases con 29 métodos propios, para el funcionamiento tanto del algoritmo genético.

Estas clases y funciones se han definido en el Apéndice A, donde se define el uso de cada función y parámetro, así como los parámetros que usan.

3.2.5 Obtención del error de localización

Para estimar el error en cada punto de la sala, en este trabajo se usa una versión modificada para su uso para cualquier tipo de sala del sistema de localización basado en SRP desarrollado por Jose Velasco en [17].

3.2.5.1 Definición de la rejilla de puntos

Para el cálculo del mapa de error para cada punto de la sala en este trabajo, aunque el sistema es capaz de usar cualquier tamaño de rejilla de puntos, se ha optado por una separación entre puntos de la rejilla de 30cm, ya que el error de posicionamiento obtenido con este sistema en [17], se estimaba en 80cm, por lo que no es necesario hacer la separación mucho más grande que el error medio.

Al tener salas tridimensionales es necesario definir la altura de la rejilla. En este trabajo se ha optado por utilizar una altura de 1.70m, que es la media de altura de países de la OCDE, aunque podría usarse cualquier valor de altura.

Este proceso se realiza mediante la función *calc_grid_points* (Tabla A.34), y la Figura 3.4 muestra un ejemplo de solución del calculo de estos puntos.

3.2.5.2 Cálculo de puntos imagen

Una vez obtenida la rejilla de puntos a analizar, para los casos que sea necesario tener en cuenta los rebotes y considerando que en este trabajo se utiliza un modelo fuente-imagen basado en el descrito por Ruth Perez en [1].

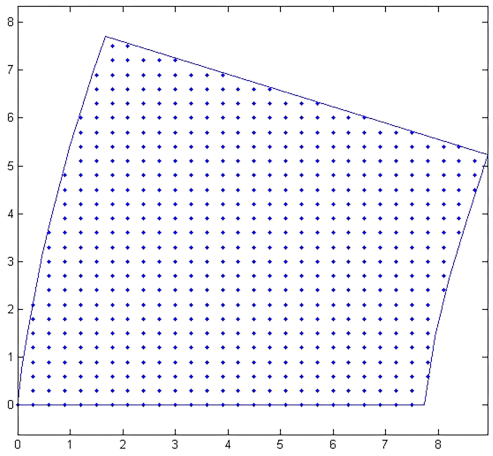


Figura 3.4: Ejemplo de cálculo de la rejilla de puntos para la sala ISPACE.

Este proceso se realiza mediante la función *calculate_IM_grid* (Tabla A.40), que obtiene de forma matemática, resolviendo los sistemas de ecuaciones de forma matricial, los puntos simétricos para cada punto de la rejilla con respecto a todas las superficies a tener en cuenta.

En la Figura 3.5 un ejemplo de cálculo de puntos imagen para una sala genérica cuadrada con una columna en el centro. En cada una de las gráficas representadas en la figura se muestran los puntos imagen obtenidos para una doble rejilla (una en el suelo y otra en el techo) para una sala cuadrada con una columna en el centro de la sala. Cada gráfica muestra los puntos imagen viables para la posición de un sensor (cuadrados rojos) de la sala (uno en cada esquina). Los puntos azul oscuro representan las rejillas y los de cada color distinto corresponden a los puntos imagen respecto a cada pared o techo.

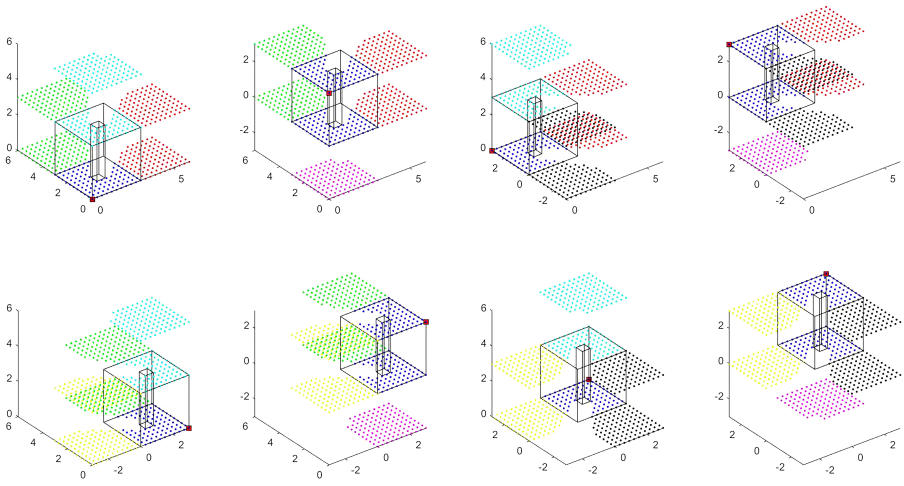


Figura 3.5: Ejemplo de cálculo puntos imagen.

3.2.6 Cálculo de zonas ciegas

No todos los puntos obtenidos después del cálculo de puntos imagen son válidos, ya que dependiendo de la posición de los micrófonos es probable que ciertos puntos de la rejilla no sean visibles o ciertos rebotes no puedan ser viables.

Debido a que la validez de los puntos a tener en cuenta depende de la posición de los micrófonos, es necesario hacer este cálculo para cada solución generada. Este proceso tiene un alto coste computacional. Por ello en este trabajo, dado que durante el proceso del posicionamiento óptimo de los micrófonos se genera una gran cantidad de soluciones, se ha optado por intentar reducir esa carga computacional repetitiva lo máximo posible.

La solución propuesta para esto en este trabajo es pre-calcular, en una primera etapa, las zonas ciegas para cada punto de la rejilla. De esta forma, en una segunda etapa, que es la que se repite para cada solución, únicamente es necesario comprobar si los micrófonos están ubicados en alguna de las zonas ciegas obtenidas previamente.

Para el calculo de las zonas ciegas, para cada punto de la rejilla (incluidos los puntos imagen), se recorren todas las superficies de dos en dos obteniendo las proyecciones de una superficie con la otra en relación a cada punto.

Este proceso se realiza mediante la función *calc_blind_spots* (Tabla A.41) que obtiene dichas proyecciones de las superficies de forma matemática, resolviendo los sistemas de ecuaciones de forma matricial para reducir el coste computacional.

La Figura 3.6 muestra un ejemplo de solución del calculo de estas zonas ciegas para un punto concreto de la sala. En la figura el cuadrado rojo representa la posición del sensor para el que se calculan las zonas ciegas, y las zonas coloreadas en cyan, rojo y rosa representan las zonas ciegas para esa posición en las paredes, techo y suelo de la sala provocadas por la columna central de la sala.

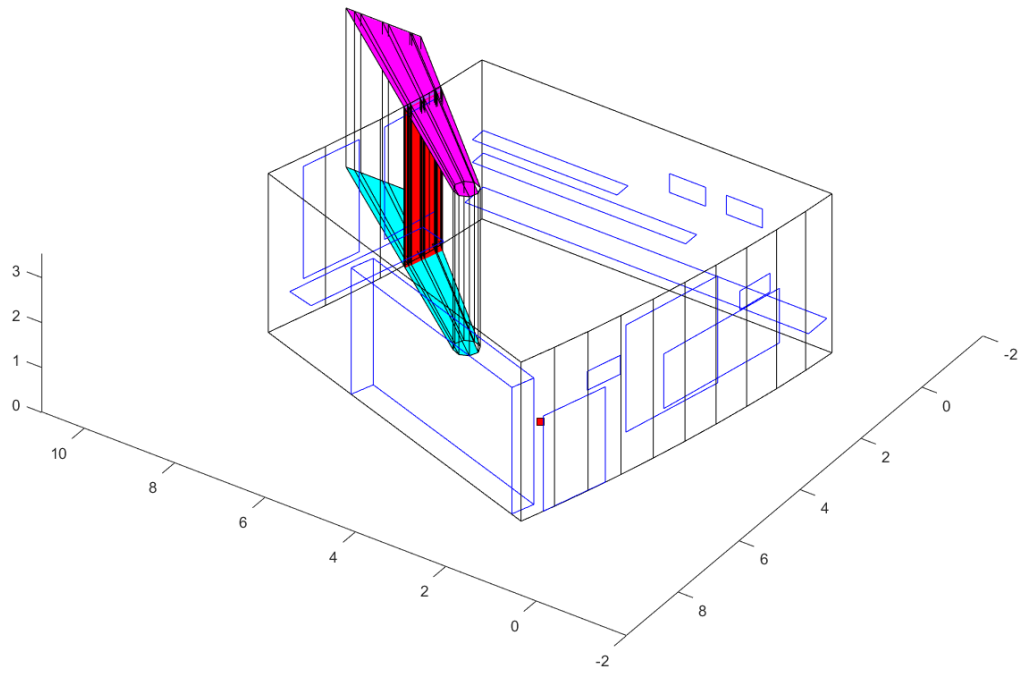


Figura 3.6: Zonas ciegas para un punto de la rejilla teniendo en cuenta los obstáculos fijos.

La segunda etapa de validación de puntos viables se realiza mediante la función *Checkpoints* (Tabla A.42), que consiste en buscar el la ubicación de cada micrófono entre todas las zonas ciegas posibles, generando la matriz de viabilidad para cada punto y cada micrófono.

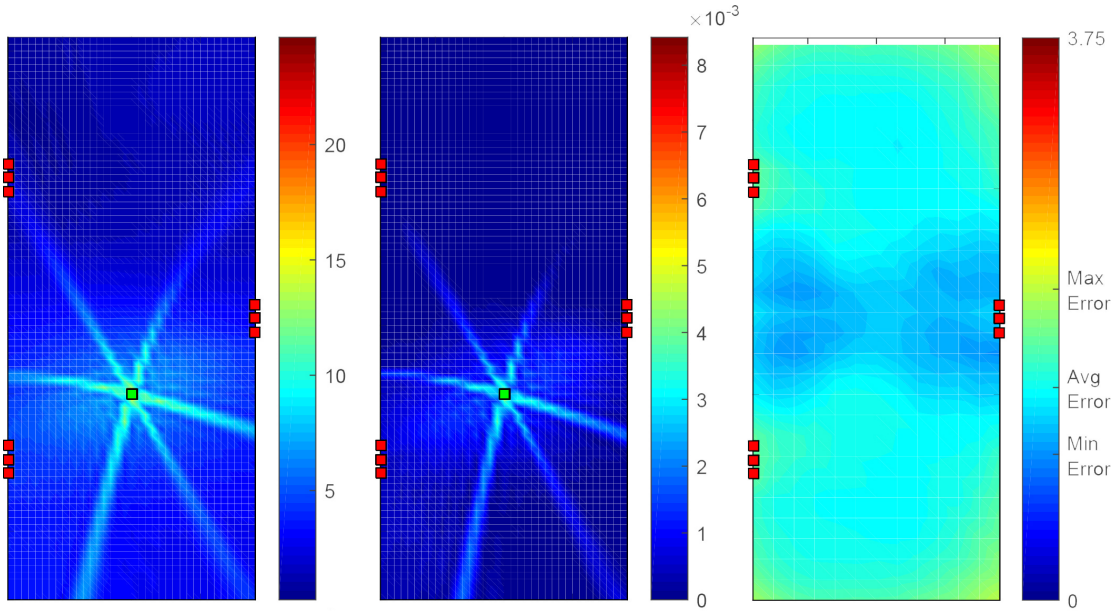
3.2.7 Cálculo SRP, SLF y error de localización

El patrón SRP para cada punto obtenido previamente se obtiene mediante la función *SRPpattern_cuboidROOM* (Tabla A.26), que es una función modificada de [17] para su uso para todo tipo de salas con y sin obstáculos.

Una vez obtenido el patrón SRP, para el cálculo del error de localización en cada punto, se convierte ese patrón SRP en la SLF, que proporciona la probabilidad de que el hablante se encuentre en cada punto teniendo en cuenta la potencia acústica en cada punto.

A partir de la SLF se calcula el error medio de localización para cada punto de la rejilla, como la contribución de cada posible localización de la fuente. Todo esto se realiza mediante la función *testsolution* (Tabla A.44).

En la Figura 3.7 se muestran dos ejemplos de la conversión del patrón SRP (Figura 3.7a), en la SLF (Figura 3.7b) y el mapa de error medio de localización para cada punto de la sala (Figura 3.7b), para 3 arrays en una sala rectangular.



(a) SRP, para un punto concreto a una altura determinada.

(b) SLF, para un punto concreto a una altura determinada.

(c) Error para cada punto.

Figura 3.7: Ejemplo de SRP, SLF y función de error.

3.3 Obtención de la ubicación óptima de los sensores

La obtención de la ubicación óptima de los sensores se realiza mediante un algoritmo genético que minimiza una función objetivo. En concreto en este trabajo se emplea una versión mejorada del algoritmo genético mono-objetivo utilizado en los trabajos anteriores [15] y [16] así como una versión multi-objetivo modificada del mismo.

El algoritmo genético de ubicación de sensores está adaptado a partir de un algoritmo genético genérico como el definido en el Capítulo 2, como se puede ver en el diagrama de la Figura 3.8.

1. Se define de la mejor forma posible la población.

   Como se busca posicionar los sensores en las superficies de las paredes, techo y/o suelo, con el fin de limitar mejor esta restricción y con la intención de reducir la dificultad del problema de tres a dos dimensiones, en este trabajo se transforma el espacio de ubicaciones posibles tridimensional (como el de la Figura 3.9a) en uno bidimensional de forma que ubique los sensores en una superficie plana formada por el desarrollo 2D de la sala (como el de la Figura 3.9b). De esta forma, los individuos se definen como las coordenadas de los sensores en ese desarrollo 2D del conjunto de sensores a posicionar, concatenadas en un vector de tamaño $1x2N$, siendo N el número de sensores para el caso de sensores aislados. Por ejemplo para un caso con 4 sensores uno de los individuos se definiría como $[x_1\ y_1\ x_2\ y_2\ x_3\ y_3\ x_4\ y_4]$.

   Cuando se utilizan arrays de micrófonos, se puede realizar el mismo proceso que para un conjunto de micrófonos aislados, con la salvedad de que para un array (o un conjunto de arrays) de micrófonos, éstos están colocados de una forma determinada.

   Para el caso de arrays de micrófonos, moviendo únicamente uno de los micrófonos (en este trabajo el micrófono central), y calculando posteriormente para la estimación del error los otros tres micrófonos, no sólo se reduce el tiempo de proceso del algoritmo genético, ya que de otra forma las soluciones que no están ubicadas con esa topología tendrían que descartarse, sino que también es posible que para obtener una solución válida y óptima es posible que no convergiera el algoritmo. Por lo que los individuos serían de tamaño $1x2A$, siendo A el número de arrays de micrófonos.

   De esta forma la población se define como el conjunto de individuos, es decir, como el conjunto de vectores definidos anteriormente. Obteniendo una matriz de tamaño $Px2N$ o $Px2A$, siendo P el tamaño de la población.

2. Se define el proceso de inicialización en el cual pueden darse dos opciones, que ya estuviera creada la población inicial o que no estuviera creada. Si estuviera creada se salta el resto de este proceso y en el caso de que no estuviera se crea la población inicial de forma aleatoria cumpliendo los requisitos del problema.

   Para ello se define una rejilla de puntos muy pequeña de forma que haya muchos puntos posibles y se cogen PxN puntos y se colocan sus coordenadas en una matriz de tamaño $Px2N$.

   Este proceso se realiza mediante las funciones *ga_initialize* (Tabla A.54) y *ga_create_population* (Tabla A.56), que calcula un conjunto de puntos posibles en base a los criterios deseados, como por ejemplo el de la Figura 3.10, y luego de forma aleatoria selecciona las posiciones los individuos de la generación inicial.

3. Se define la función de cruce.

   Las funciones de cruce usan dos soluciones (padres) para generar una tercera (hijo) dependiente de éstas, ya sea obteniendo el punto central de ambas (Cruce por punto intermedio) o mezclando la información de ambas (Cruce por uno o dos puntos). Ésto se realiza con el fin de que la tercera solución, como sucede en la naturaleza donde los hijos heredan información de los padres, pueda ser una solución mejor que las anteriores.

   Dependiendo del tipo de cruce puede darse el caso de que al generar nuevos individuos éstos no cumplan los requisitos del problema por lo que se puede aplicar o no corrección de dichos individuos para que vuelvan a cumplir los requisitos. En este trabajo dicha corrección (en el caso de que se

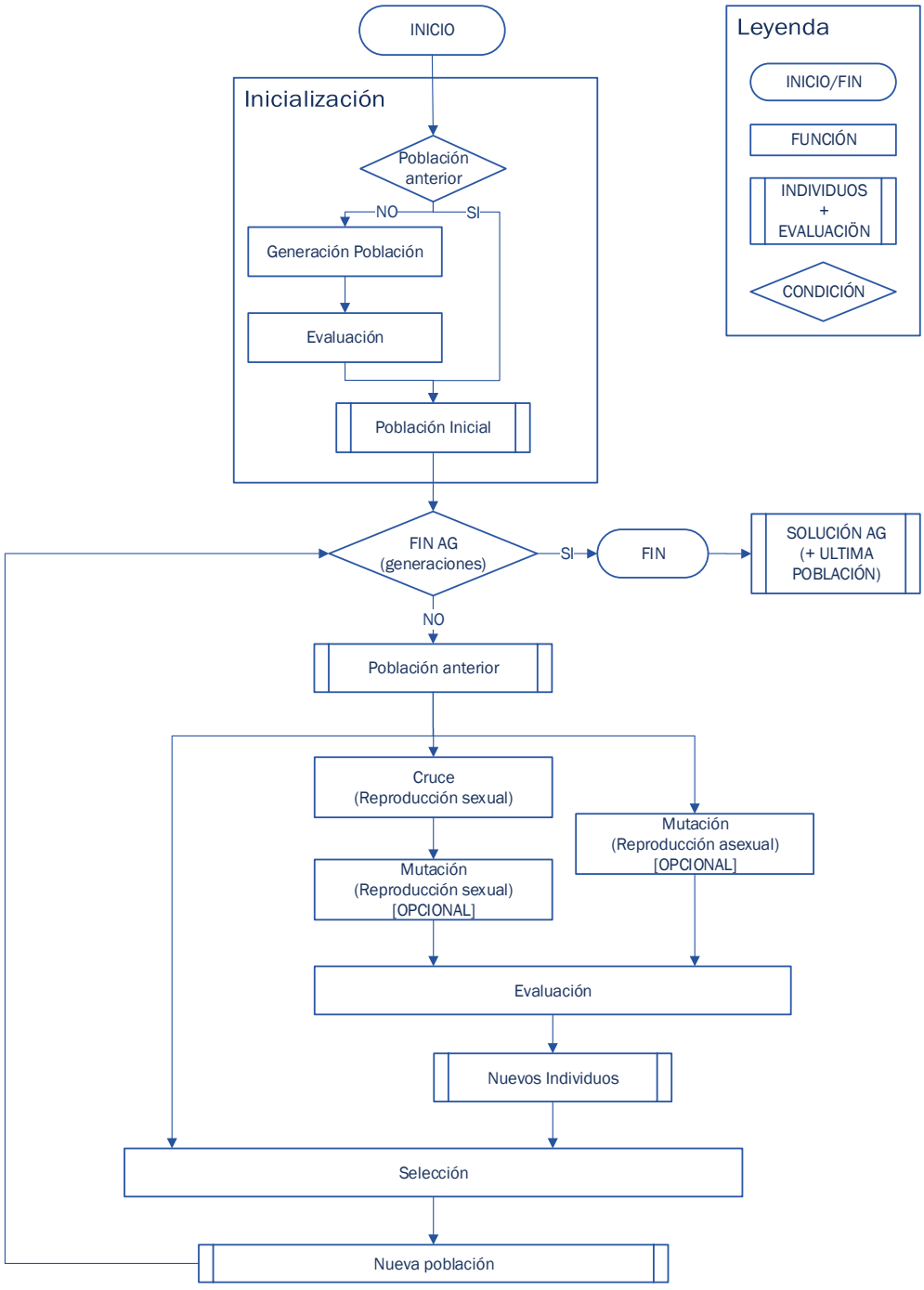
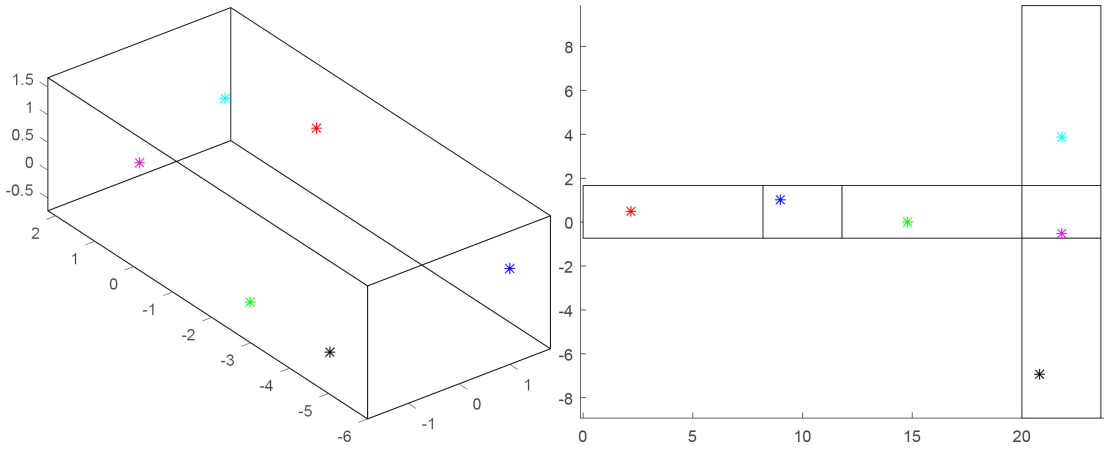


Figura 3.8: Diagrama de bloques del algoritmo genético empleado en este trabajo.

seleccione) se realiza eliminando los sensores que no cumplen los requisitos y sustituyéndolos de forma aleatoria por uno que si los cumpla. De esta forma no solo se evita perder soluciones que pueden ser buenas, si no que además se genera una nueva aleatoriedad que puede evitar caer en óptimos locales.



(a) Puntos en espacio tridimensional. (b) Puntos en espacio bidimensional equivalente.

Figura 3.9: Ejemplo de conversión de puntos.

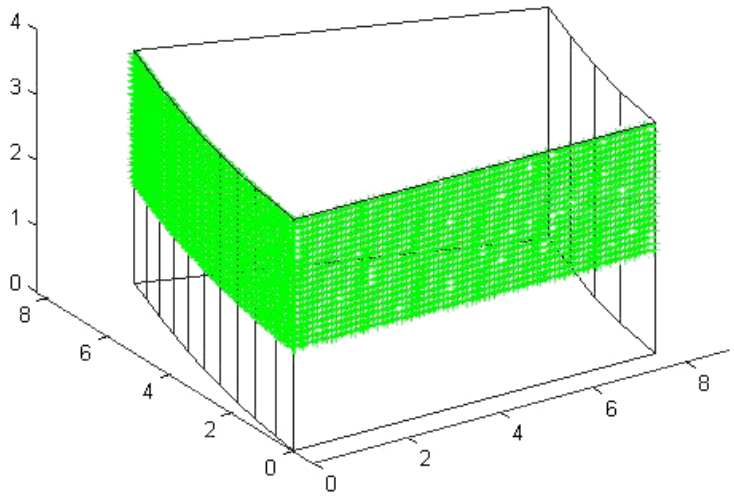


Figura 3.10: Ejemplos de puntos posibles para las soluciones, limitando las posiciones posibles a una altura fija y evitando dos de las paredes posibles.

En el sistema desarrollado por medio de la interfaz (o manipulando manualmente los datos) se puede seleccionar cualquiera de estos tres tipos de cruce.

Este proceso se realiza mediante la función *ga_crossover* (Tabla A.51).

4. Se define la función de mutación. Las funciones de mutación en cambio, parten de una solución y la modifican de forma aleatoria, al igual que en la naturaleza, eso se consigue modificando uno o varios puntos de una solución generando una solución mutada que varíe aleatoriamente respecto a la anterior.

   Además imitando de nuevo los procesos naturales donde existen dos tipos de reproducción sexual o asexual, que pueden darse de forma simultánea, en este trabajo la mutación puede seleccionarse teniendo en cuenta esta posibilidad. De esta forma si quiere emularse un proceso natural de reproducción sexual los hijos generados en el proceso de cruce podrían de forma aleatoria mutar, en

cambio si se quiere emular un proceso de reproducción asexual sería un porcentaje de la población previa la que podría mutar aleatoriamente generando unos nuevos hijos-mutados.

En la mutación al igual que en el cruce es posible que los individuos generados no cumplan los requisitos del problema, por lo que si se desea pueden corregirse con el mismo método que para el cruce.

En el sistema desarrollado por medio de la interfaz (o manipulando manualmente los datos) se puede seleccionar cualquiera de los tipos de mutación.

Este proceso se realiza mediante la función *ga_mutation* (Tabla A.52).

5. Se define la función de evaluación de la población o función objetivo. Dicha función se centra en evaluar cada individuo de la población asignándole un valor para que después la función de selección pueda escoger los mejores individuos para trabajar con ellos.

   La función objetivo usada en este trabajo es la llamada *ga_evaluation* (Tabla A.55), que a su vez llama a la función *fitness_function* (Tabla A.48), que a su salida (la cual hay que optimizar) da información del error medio, máximo o mínimo para esa solución. Dicha función calcula ese error o bien mediante la CRLB para sensores con localización basada en TDOA, o bien a partir del patrón SRP para la localización con micrófonos.

   Está adaptada también para garantizar que se cumplan ciertas características, por medio de la función *is_valid_solution* (Tabla A.49), y si no penalizar la solución. Esta penalización se aplica a las soluciones que no cumplen las características de ubicación restringidas, por ejemplo que el array de micrófonos no tenga todos los micrófonos en la misma pared, que no cumpla los requisitos de altura mínima o máxima o que no estén situados dentro de las superficies deseadas.

   En este trabajo se tienen en cuenta los obstáculos fijos de la sala, como las columnas. Por tanto hay que tenerlos en cuenta en tanto para los cálculos del TDOA (en la matriz de covarianza y en el cálculo del jacobiano), como para los cálculos de SRP (para calcular los caminos válidos). Para esto se considera que en caso de que un obstáculo esté situado entre el objetivo y un sensor, la distancia entre ellos se considera infinita.

   Para calcular si hay un obstáculo en la trayectoria situada entre un sensor y el punto objetivo, se traza una línea que une estos dos puntos y se comprueba si alguno de los puntos de ésta están situados dentro del polígono formado por el obstáculo en el suelo.

6. Por último se define la función de selección, que se realiza de forma diferente según se quiera usar un algoritmo genético mono-objetivo o multi-objetivo. Este proceso se realiza mediante la función *ga_selection* (Tabla A.53).

   - Selección mono-objetivo: En este trabajo la selección para un algoritmo genético mono-objetivo se ha definido mediante *ranking* exclusivamente ya que funciona correctamente para este tipo de problemas.
   - Selección multi-objetivo: Para la selección multi-objetivo se ha tenido en cuenta el proceso de selección de NSGA-II [3], el cual se basa en ordenar las soluciones en distintos frentes de pareto no dominados mediante la función *non_dominated_short* (Tabla A.57) y ordenando las soluciones de cada pareto en base a su distancia *crowding* mediante la función *crowding_distance* (Tabla A.58). En la Figura 3.11 se muestra un ejemplo de ordenación de soluciones en frentes de pareto no dominados, donde cada color representa un frente de pareto no dominado.

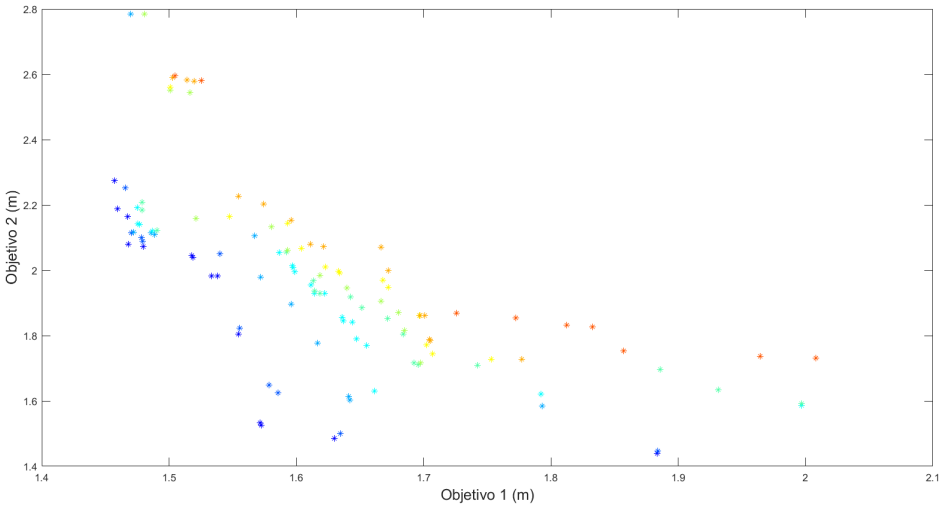


Figura 3.11: Ejemplo de ordenación de soluciones multi-objetivo.

Capítulo 4

Resultados

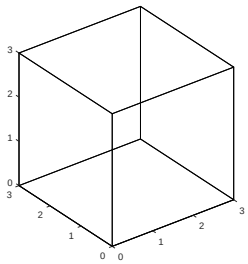
Si no sabes cómo hacer la pregunta correcta, no descubrirás nada.

W. Edward Deming

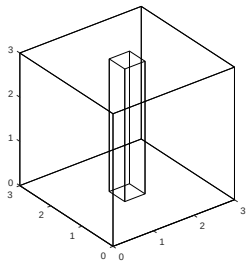
4.1 Introducción

A lo largo de este capítulo se realiza un estudio cualitativo y cuantitativo del funcionamiento del sistema de posicionamiento óptimo de sensores desarrollado descrito en el Capítulo 3, a través de un conjunto de experimentos.

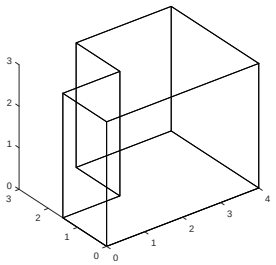
Para los experimentos se han usado varias salas, tanto teóricas como reales simuladas. Las salas teóricas se han utilizado en su mayoría para demostrar el funcionamiento del algoritmo, se ha usado una sala cuadrada de 3x3x3m con y sin obstáculos (representadas las Figuras 4.1a y 4.1b respectivamente) y una sala de forma arbitraria con forma de L que cabe en un espacio paralelepípedo de 4x3x3m (representada en la Figura 4.1c. En cuanto al funcionamiento en salas reales, con el fin de poder corroborar la mejora de la solución obtenida en este trabajo con resultados reales en proyectos futuros, se han simulado dos espacios inteligente reales, como son IDIAP (fotografiada en la Figura 4.2a), una sala rectangular de 3.6x8.2x2.4m, sin obstáculos con la que podemos contrastar datos reales de micrófonos (representada en la Figura 4.2d) e ISPACE (fotografiada en las Figuras 4.2b y 4.2c), una sala con forma irregular que cabe en un espacio paralelepípedo de 9x8x3.6m, con paredes curvas y con obstáculos, la cual se va a simular tanto sin obstáculos como con obstáculos (representadas respectivamente en las Figuras 4.2e y 4.2f).



(a) Sala cuadrada sin obstáculos.

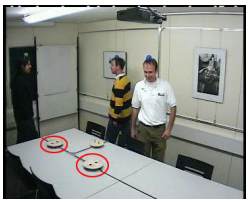


(b) Sala cuadrada con obstáculos.



(c) Sala de forma arbitraria con forma de L.

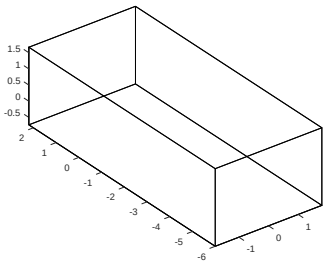
Figura 4.1: Salas teóricas.



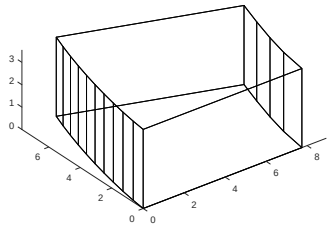
(a) Fotografía del espacio Inteligente IDIAP.

(b) Fotografía del espacio inteligente ISPACE.

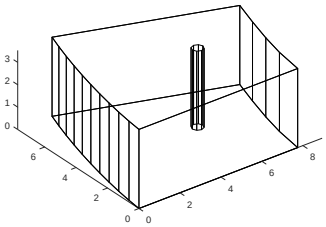
(c) Fotografía del espacio inteligente ISPACE.



(d) Espacio Inteligente IDIAP.



(e) Espacio inteligente ISPACE sin tener en cuenta la columna.



(f) Espacio inteligente ISPACE teniendo en cuenta la columna.

Figura 4.2: Salas reales.

En cuanto al tipo de sensores que se van a simular, se utilizan sensores genéricos usando localización basada en TDOA y micrófonos utilizando localización basada en SRP. Para ambos tipos de sensores además se permiten 2 tipos de configuración de posicionamiento, ya sean todos los sensores posicionados de forma independiente o formando agrupaciones de micrófonos en forma de T invertida como el de la Figura 2.6, a los que en este trabajo se les llama arrays de micrófonos.

4.2 Comprobación del funcionamiento del algoritmo

Con el fin de comprobar el funcionamiento correcto del algoritmo se comenzó evaluándolo para un caso sencillo usando sensores genéricos (mediante TDOA) para una sala cuadrada con 8 sensores a lo largo de 200 generaciones.

Como se puede ver en la gráfica de la figura 4.3 según va avanzando el algoritmo genético el error medio va disminuyendo hasta que llega a estabilizarse, por lo que podemos decir que el algoritmo converge a partir de ese número de generaciones para la configuración del algoritmo genético del ejemplo. Ésto no significa que sea la mejor ubicación posible ya que ampliando el número de generaciones del algoritmo y la población de éste la solución puede mejorar. El error mostrado se calcula a una altura de 1,7 m teniendo en cuenta la estatura media humana, ya que es aproximadamente la altura donde se producen los sonidos.

Por tanto se puede decir que el algoritmo utilizado minimiza el error medio en un tiempo razonable muy inferior al tiempo que se tardaría en probar todas las posibles soluciones, aunque no garantiza la mejor ubicación absoluta, ya que no usa un método determinista como sería probar las infinitas posibilidades.

En la figura 4.4 se puede ver el funcionamiento de dicho algoritmo de forma visual a lo largo de las 200 generaciones. En estas figuras se representa cómo evoluciona el posicionamiento óptimo por intervalos de generaciones, así como el mapa del error de localización para cada resultado.

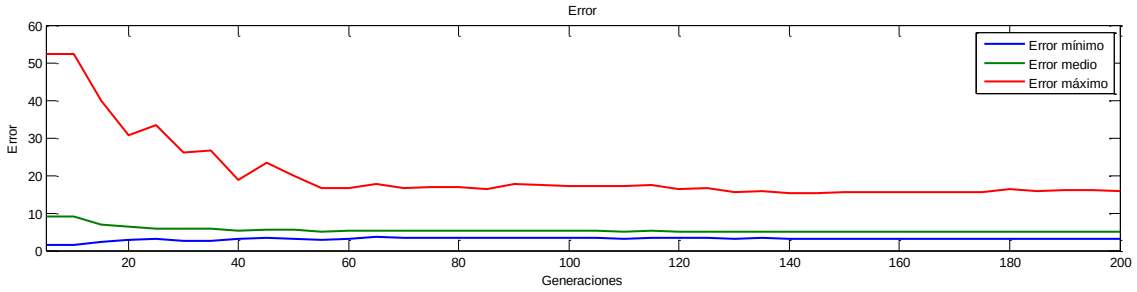
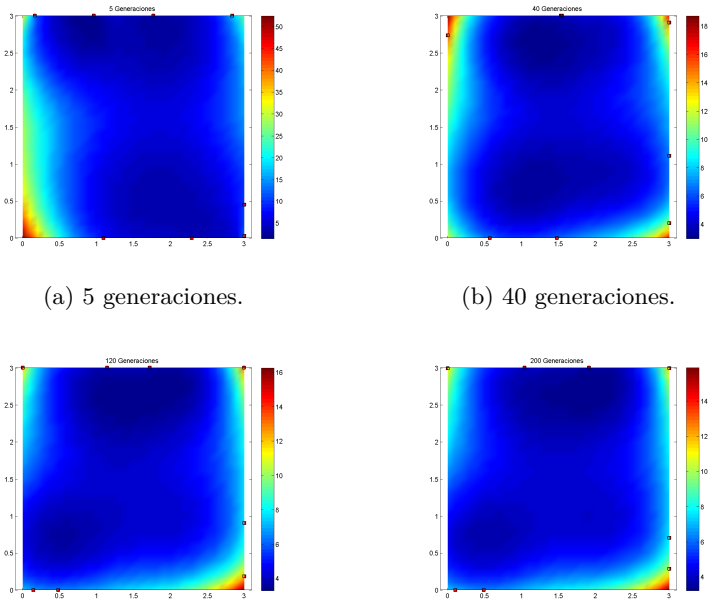


Figura 4.3: Evolución del algoritmo genético a lo largo de 200 generaciones



(a) 5 generaciones.

(b) 40 generaciones.

(c) 120 generaciones.

(d) 200 generaciones.

Figura 4.4: Error para 8 sensores a lo largo 200 generaciones.

4.3 Resultados utilizando posicionamiento basado en TDOA

A continuación, se muestran varios resultados obtenidos con el algoritmo genético usado, para el uso sensores IR con y sin obstáculos sin tener en cuenta el multicamino basado en TDOA.

En el apartado anterior 4.2 se ha mostrado el resultado obtenido para una sala cúbica de 3 m de lado, con 8 sensores con localización basada en TDOA y sin obstáculos. En ese mismo contexto localización basada en TDOA, se encuentran los siguientes experimentos.

En la figura 4.5 se puede ver la solución óptima calculada para la misma sala cúbica la figura 4.4 pero con una columna cuadrada en el centro de la sala.

El la figura 4.6 se puede ver la posibilidad de análisis para una sala con obstáculos (ISPACE), con localización basada en TDOA con 10 sensores, si se tienen o no en cuenta los obstáculos. Ambas pruebas se han realizado limitando las paredes posibles para el posicionamiento de los sensores a las dos paredes donde están ubicados, ya que en la sala real en esas dos paredes no pueden ubicarse al haber ventanas y armarios en éstas.

Como se puede apreciar en ambas gráficas, el error máximo de localización se encuentra en la esquina superior derecha ya que es el punto más alejado a todos los sensores y el error es dependiente de la distancia al cuadrado. Si se compara con la sala cuadrada el error es mucho mayor por la misma razón,

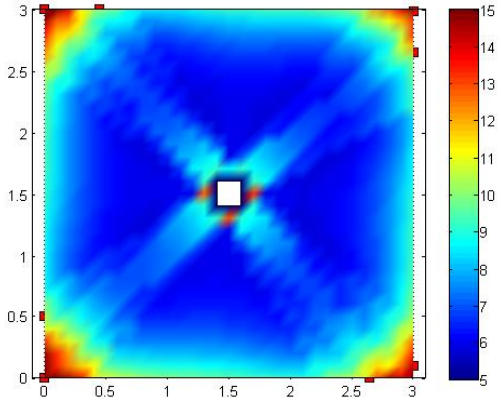
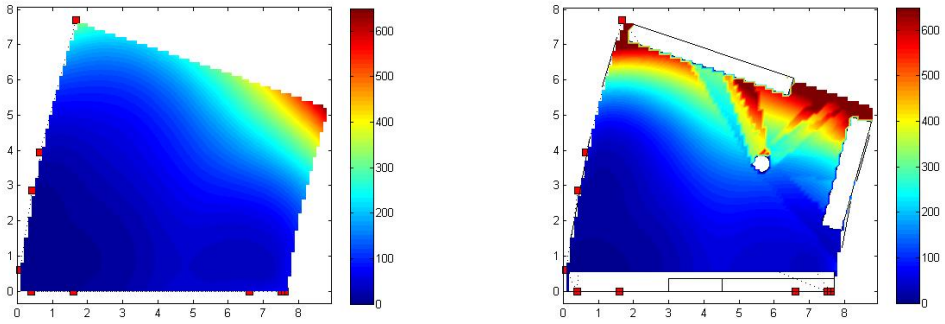


Figura 4.5: Localización con 8 sensores basada en TDOA con un obstáculo central



(a) Posicionamiento óptimo sin obstáculos.

(b) Posicionamiento óptimo con obstáculos.

Figura 4.6: Contraste de solución sin obstáculos con solución con obstáculos.

ya que las paredes son más de dos veces más largas. Para el análisis teniendo en cuenta los obstáculos como se puede apreciar aparecen zonas con mayor error detrás de la columna de la sala, lo cual coincide con lo que se podría esperar en la realidad. El error máximo al tener en cuenta los obstáculos aumenta al tener en cuenta la penalización por obstáculo, por lo que en la gráfica aparece una zona con un valor mayor que el máximo valor representado (zona rojo oscuro).

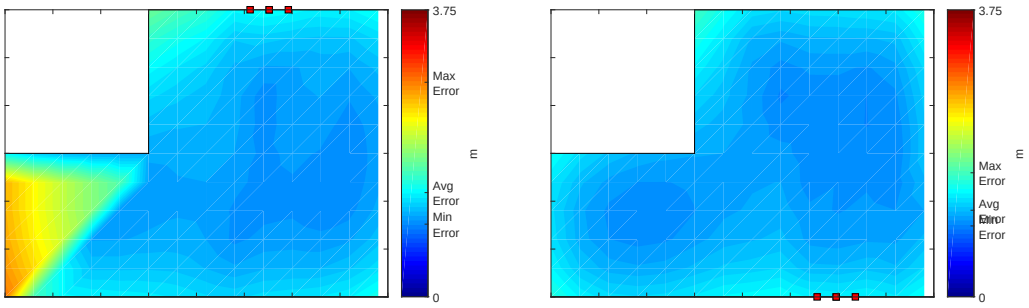
Como se puede ver en las distintas figuras expuestas el algoritmo de ubicación de sensores funciona correctamente para este tipo de posicionamiento basado en TDOA. Sin embargo, en este trabajo no se hace mucho hincapié en este método de posicionamiento, ya que ha sido desarrollado en profundidad en [26], con la salvedad de probarlo en otro tipo de salas no rectangulares.

4.4 Resultados utilizando posicionamiento basado en SRP

4.4.1 Resultados previos

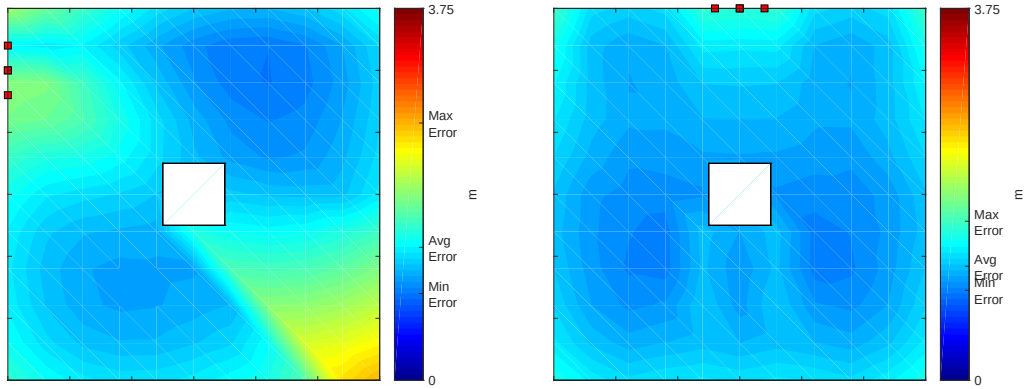
Para comprobar el funcionamiento del sistema desarrollado para posicionamiento basado en SRP para todo tipo de salas se ha testeado el algoritmo para una una sala cuadrada con una columna en el medio y para una sala de forma arbitraria, para minimizar el error medio.

En las Figuras 4.7a y 4.8a se representa el posicionamiento aleatorio de 1 array de micrófonos para ambas salas y en las Figuras 4.7b y 4.8b se muestra el posicionamiento óptimo con el algoritmo genético diseñado.



(a) Posicionamiento aleatorio. (b) Posicionamiento óptimo.

Figura 4.7: Sala con forma arbitraria.



(a) Posicionamiento aleatorio. (b) Posicionamiento óptimo.

Figura 4.8: Sala cuadrada con obstáculos.

Como se puede ver el algoritmo minimiza considerablemente el error medio, por lo que se asume que el algoritmo funciona correctamente, para cualquier tipo de salas, por lo que se pasa a un análisis exhaustivo del algoritmo para los espacios inteligentes reales de IDIAP e ISPACE.

4.4.2 Estimación de los parámetros óptimos del algoritmo genético

Teniendo en cuenta el tiempo de simulación de este tipo de posicionamiento, y para asegurar la correcta interpretación de los resultados, se ha procedido a estimar el tamaño de población y el número de generaciones óptimo para este problema.

Para estimar los parámetros óptimos de tamaño de población y número de generaciones se han realizado experimentos para la sala IDIAP, con 8 micrófonos (y 2 arrays de micrófonos) en un intervalo de 30 a 300 individuos (con intervalos de 30 individuos), durante 3000 generaciones. En la Figura 4.9 se representa el error medio de localización para el posicionamiento óptimo de los experimentos con 2 arrays de micrófonos (y en la Figura 4.10 para 8 micrófonos posicionados de forma independiente), para cada configuración del tamaño de la población a lo largo de las 3000 generaciones.

Para el caso de micrófonos independientes al tener mucha más libertad de movimientos la convergencia se alcanza con un número de generaciones mucho mayor, sin embargo como se puede observar, para ambos casos los mínimos se alcanzan para un número de generaciones de 500 con una precisión suficiente, y se obtienen buenos resultados para un tamaño de población de 120 individuos.

Sin embargo para comprobar mejor dichos parámetros (sobretodo el tamaño de población), se evaluó también para la sala ISPACE con los mismos intervalos de individuos para 1000 generaciones, ya que por la complejidad de la sala el tiempo de procesado es mucho mayor. En la Figura 4.11 se representa el error

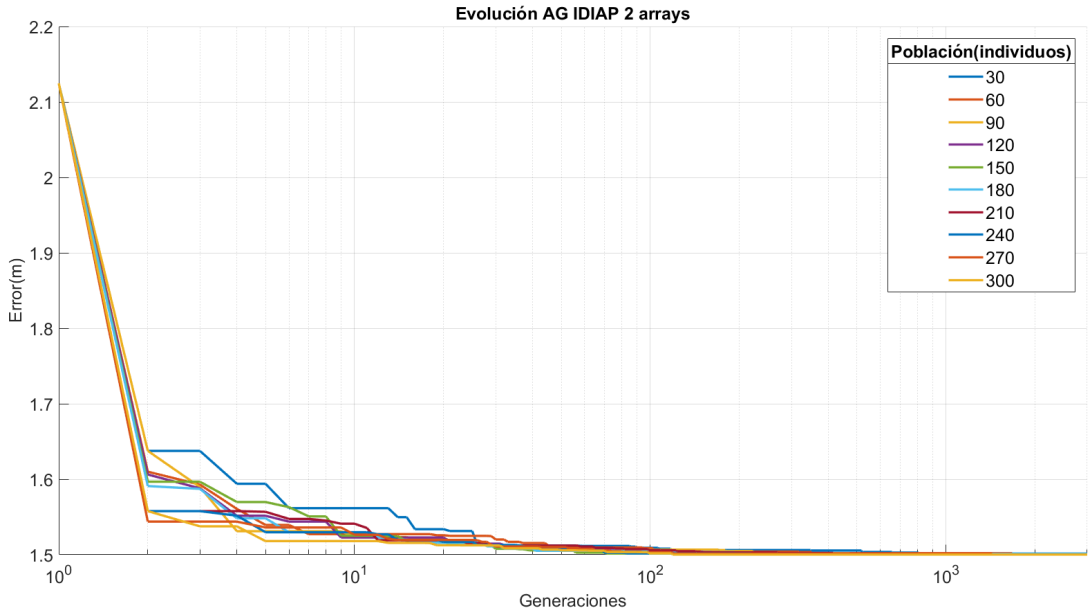


Figura 4.9: Evolución algoritmo para sala IDIAP con 2 arrays de micrófonos

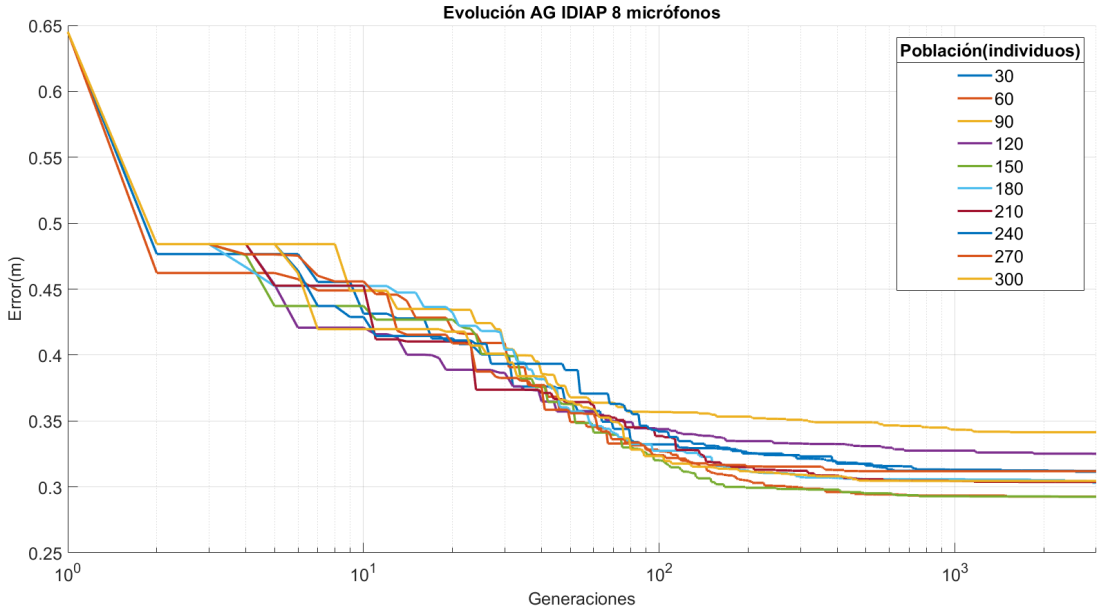


Figura 4.10: Evolución algoritmo para sala IDIAP con 8 micrófonos

medio de localización para el posicionamiento óptimo de los experimentos con 2 arrays de micrófonos (y en la Figura 4.12 para 8 micrófonos posicionados de forma independiente), para cada configuración del tamaño de la población a lo largo de las 1000 generaciones.

Como se puede observar para ambos casos se alcanza la convergencia con suficiente precisión para 500 generaciones y además como en el caso de 120 individuos establecido para la sala IDIAP se obtienen buenos resultados, por lo que no es necesario poner un número de población mayor.

Tras los experimentos realizados se concluye que los parámetros más adecuados para el problema bajo estudios son: 1000 generaciones (el doble del número de generaciones para las que se ha alcanzado la convergencia en los ejemplos), y un tamaño de población de 120, por lo que son los datos que se usarán para los siguientes resultados.

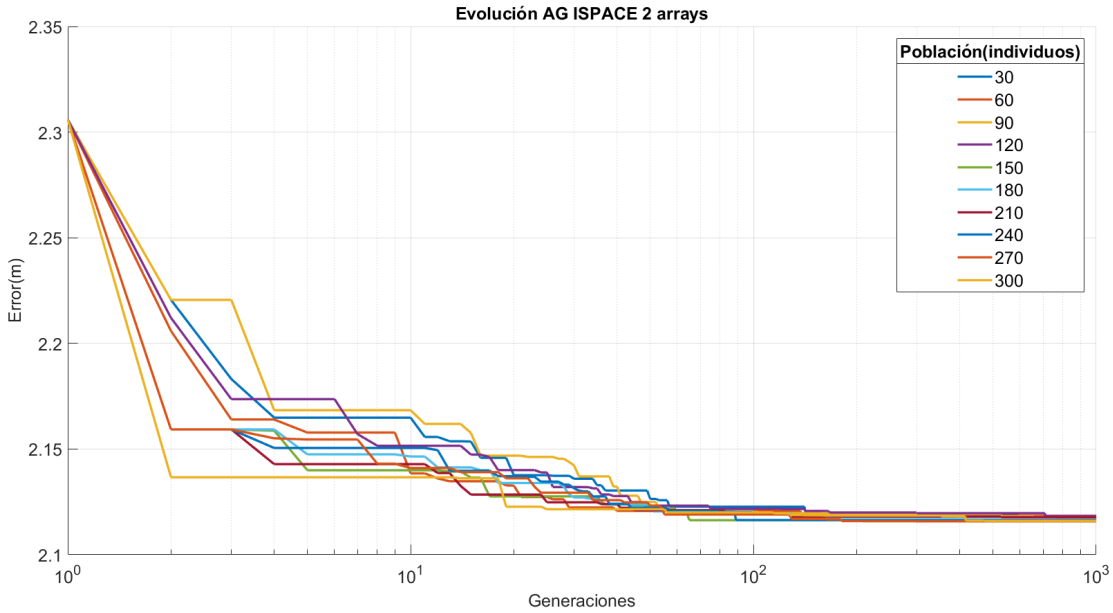


Figura 4.11: Evolución algoritmo para sala ISPACE con 2 arrays de micrófonos

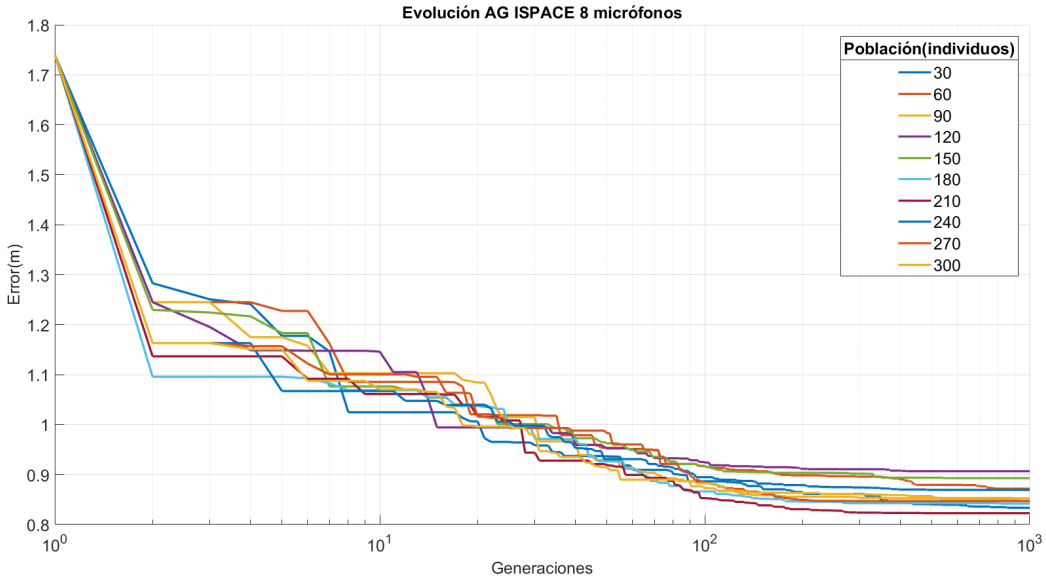


Figura 4.12: Evolución algoritmo para sala ISPACE con 8 micrófonos

4.4.3 Simulaciones mono-objetivo

En este apartado se exponen los resultados obtenidos con el sistema desarrollado en este TFM, para 3 tipos de sala:

- Sala rectangular (sala IDIAP sin obstáculos).
- Sala arbitraria sin obstáculos (sala ISPACE sin obstáculos).
- Sala arbitraria con obstáculos (sala ISPACE con obstáculos).

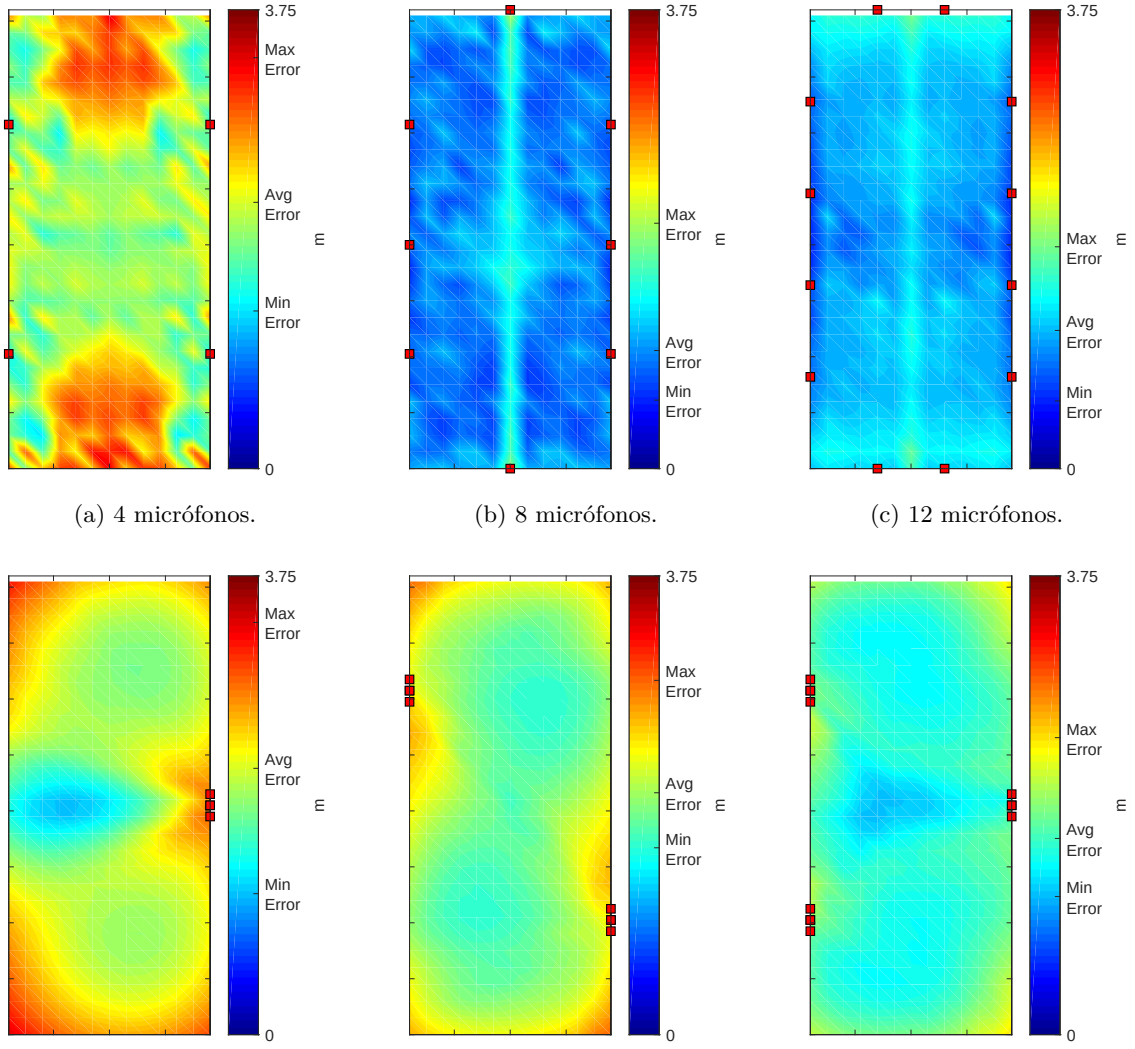
Para estos tres tipos de salas se ha realizado una comparativa visual y cuantitativa del resultado del sistema de posicionamiento óptimo obtenido comparado un posicionamiento manual de 4, 8 y 12

micrófonos independientes y de 1, 2 y 3 arrays de micrófonos. Además se ha comparado cuantitativamente también ambos resultados con el obtenido de forma aleatoria.

Todos estos experimentos se han realizado para los parámetros óptimos de 1000 generaciones y una población de 120 individuos, obtenidos en el apartado anterior.

4.4.3.1 Resultados mono-objetivo

De la Figura 4.14 a la 4.18 se muestran los mapas de error medio de localización de los resultados del algoritmo de posicionamiento desarrollado comparándolos con el posicionamiento manual de los micrófonos, mostrado en la Figura 4.13.



(a) 4 micrófonos. (b) 8 micrófonos. (c) 12 micrófonos.

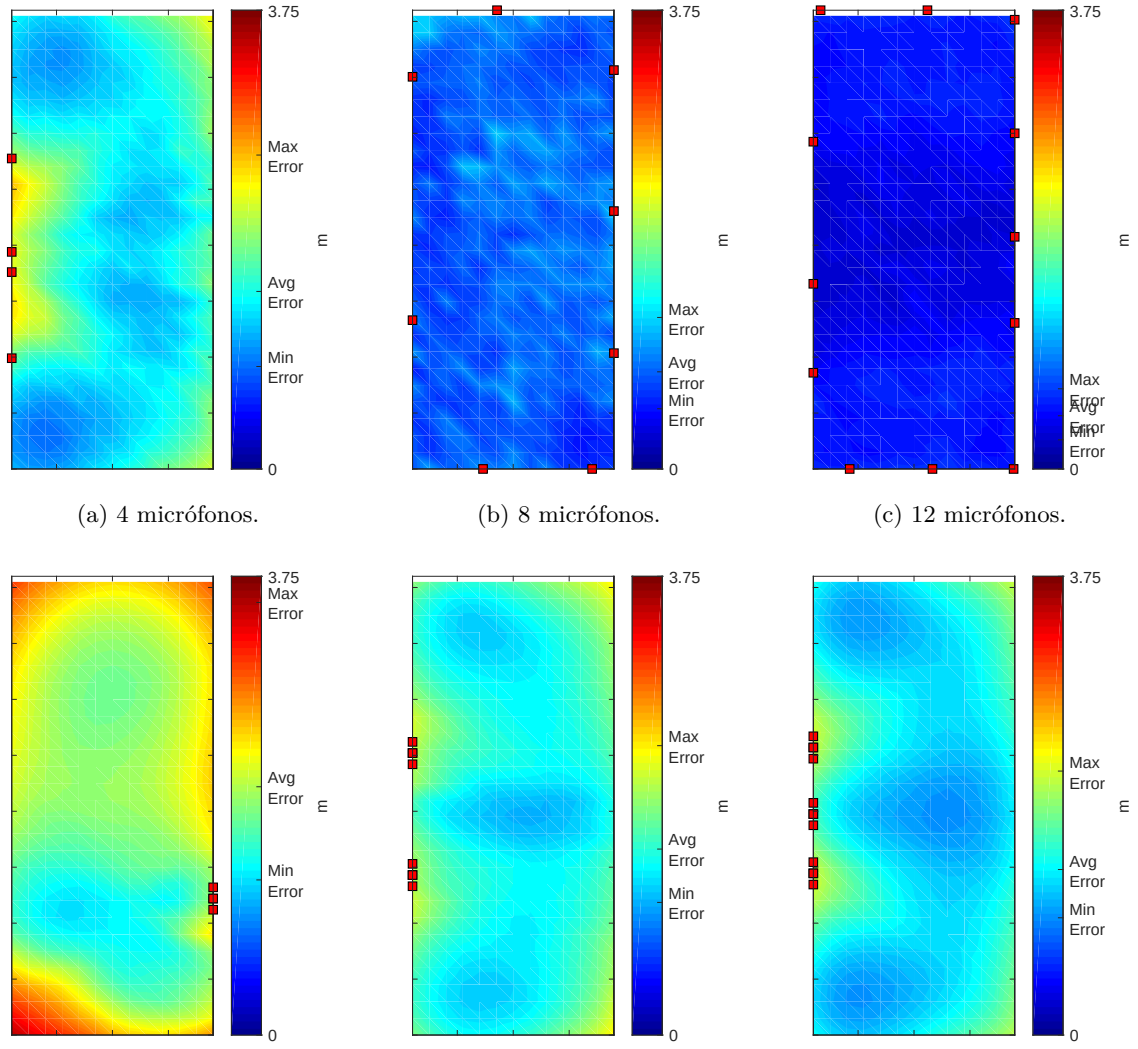
(d) 1 array (4 mics). (e) 2 arrays (8 mics). (f) 3 arrays (12 mics).

Figura 4.13: Mapa de error de localización para configuraciones manuales para la sala IDIAP: micrófonos individuales (fila superior), y arrays de micrófonos (fila inferior).

Estos mapas de error, representados como un mapa de color, representan el error medio de localización para cada punto de la sala en una rejilla bidimensional ubicada a una altura fija de 1.7m de altura, donde en la barra de colores además se representan el error medio de localización para todos los puntos de la sala, así como el error máximo y el error mínimo.

Además de la Tabla 4.1 a la 4.3 se muestra una comparación cuantitativa del error de localización medio obtenido con el sistema de posicionamiento óptimo desarrollado en este trabajo (columna *Opt*), con

respecto un posicionamiento aleatorio (columna Rnd), y con respecto al posicionamiento manual (columna Man). En las dos últimas columnas, se presenta también la mejora relativa del posicionamiento óptimo en comparación con el posicionamiento aleatorio ($\Delta_p^r = (Rnd - Opt)/Rnd$) y el posicionamiento manual ($\Delta_p^m = (Man - Opt)/Man$).



(a) 4 micrófonos. (b) 8 micrófonos. (c) 12 micrófonos.

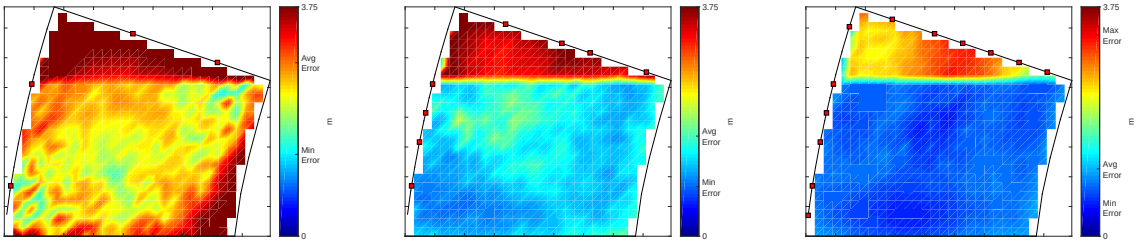
(d) 1 array (4 mics). (e) 2 arrays (8 mics). (f) 3 arrays (12 mics).

Figura 4.14: Mapa de error de localización para configuraciones óptimas para la sala IDIAP: micrófonos individuales (fila superior), y arrays de micrófonos (fila inferior).

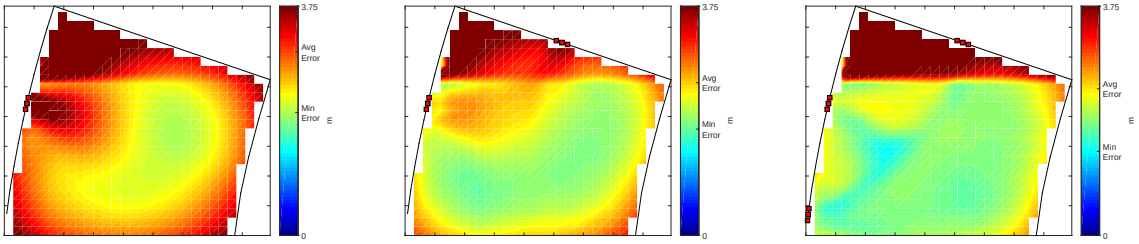
Tabla 4.1: Rectangular room: Comparación del error medio entre el posicionamiento aleatorio, el posicionamiento manual y el posicionamiento óptimo propuesto.

$\#Sensores$		Rnd	Man	Opt	Δ_p^r	Δ_p^m
4	(individual)	2.0590	2.1767	1.4513	29.51 %	33.33 %
8	(individual)	1.4298	0.9652	0.7942	44.45 %	17.72 %
12	(individual)	1.1301	1.1295	0.4351	61,50 %	61.48 %
4	(1 array)	2.4907	2.1773	2.0295	18.52 %	6.79 %
8	(2 arrays)	2.2928	1.9776	1.5202	33.70 %	23.13 %
12	(3 arrays)	1.9625	1.6042	1.3530	31.06 %	15.66 %

En las figuras y tablas anteriores se puede ver como el algoritmo de posicionamiento mono-objetivo propuesto en este trabajo funciona de la forma deseada para diferentes salas y tanto con la presencia de obstáculos como sin ellos.

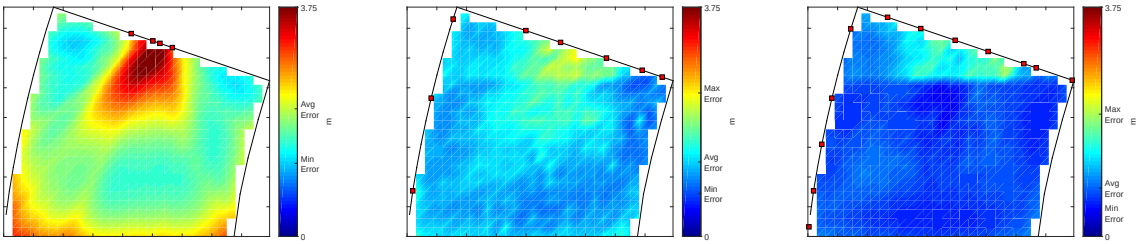


(a) 4 micrófonos. (b) 8 micrófonos. (c) 12 micrófonos.

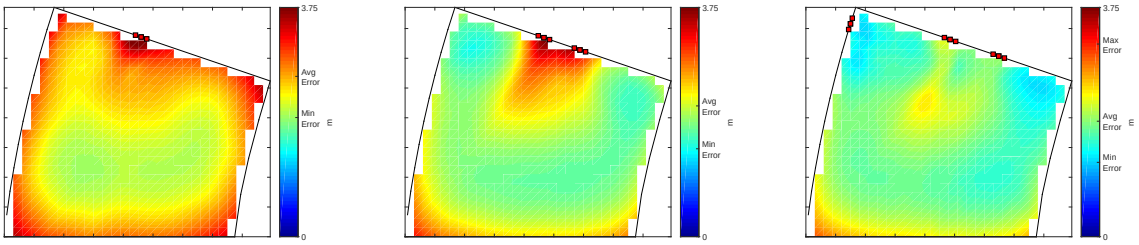


(d) 1 array (4 mics). (e) 2 arrays (8 mics). (f) 3 arrays (12 mics).

Figura 4.15: Mapa de error de localización para configuraciones manuales para la sala ISPACE sin obstáculos: micrófonos individuales (fila superior), y arrays de micrófonos (fila inferior).



(a) 4 micrófonos. (b) 8 micrófonos. (c) 12 micrófonos.

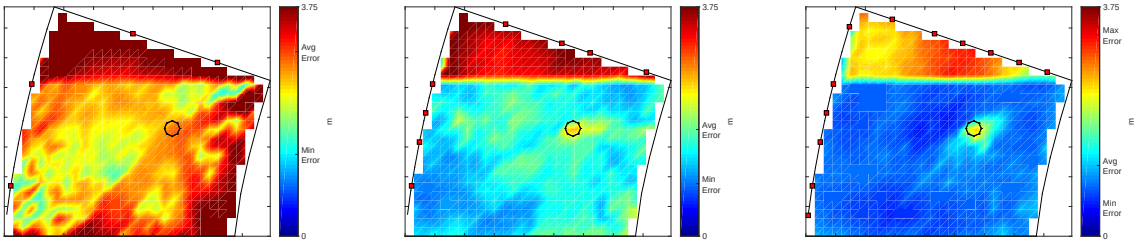


(d) 1 array (4 mics). (e) 2 arrays (8 mics). (f) 3 arrays (12 mics).

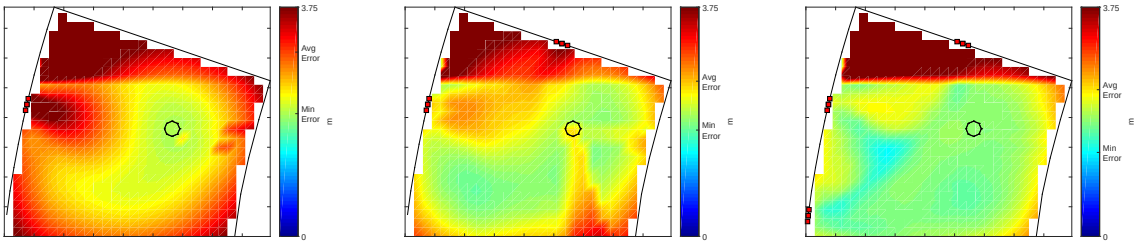
Figura 4.16: Mapa de error de localización para configuraciones óptimas para la sala ISPACE sin obstáculos: micrófonos individuales (fila superior), y arrays de micrófonos (fila inferior).

Tabla 4.2: ISPACE sin obstáculos: Comparación del error medio entre el posicionamiento aleatorio, el posicionamiento manual y el posicionamiento óptimo propuesto.

$\#Sensores$		Rnd	Man	Opt	Δ_p^r	Δ_p^m
4	(individual)	2.8653	2.8362	2.0875	27.15 %	26.40 %
8	(individual)	1.7574	1.6369	1.2180	30.70 %	25.60 %
12	(individual)	1.6866	1.1018	0.7895	53.19 %	28.34 %
4	(1 array)	3.0941	3.0065	2.6226	15.24 %	12.77 %
8	(2 arrays)	2.8486	2.4993	2.1392	24.90 %	14.41 %
12	(3 arrays)	2.6345	2.3998	1.8889	28.30 %	21.29 %

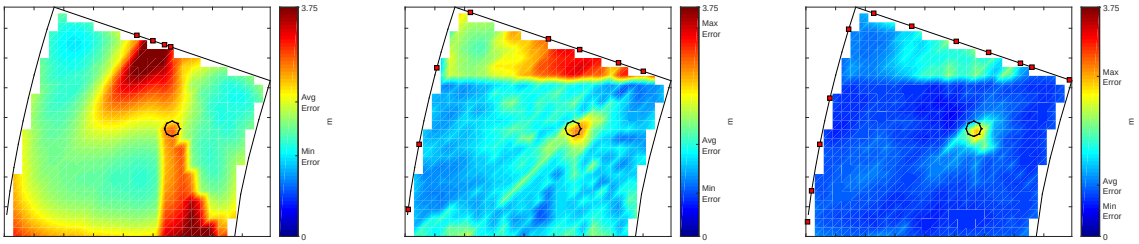


(a) 4 micrófonos. (b) 8 micrófonos. (c) 12 micrófonos.

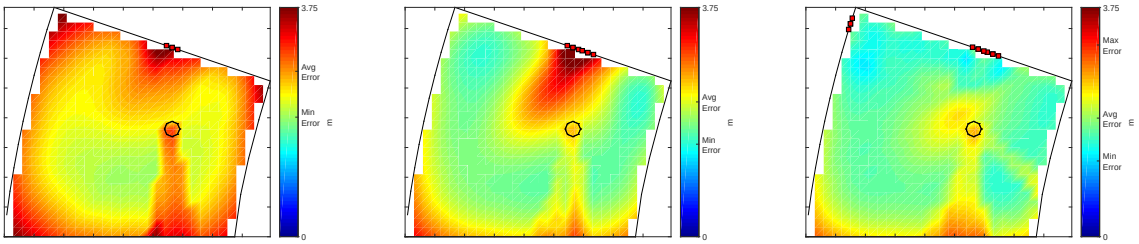


(d) 1 array (4 mics). (e) 2 arrays (8 mics). (f) 3 arrays (12 mics).

Figura 4.17: Mapa de error de localización para configuraciones manuales para la sala ISPACE con obstáculos: micrófonos individuales (fila superior), y arrays de micrófonos (fila inferior).



(a) 4 micrófonos. (b) 8 micrófonos. (c) 12 micrófonos.



(d) 1 array (4 mics). (e) 2 arrays (8 mics). (f) 3 arrays (12 mics).

Figura 4.18: Mapa de error de localización para configuraciones óptimas para la sala ISPACE con obstáculos: micrófonos individuales (fila superior), y arrays de micrófonos (fila inferior).

Tabla 4.3: ISPACE con obstáculos: Comparación del error medio entre el posicionamiento aleatorio, el posicionamiento manual y el posicionamiento óptimo propuesto.

$\#Sensores$		Rnd	Man	Opt	Δ_p^r	Δ_p^m
4	(individual)	2.9699	3.0166	2.2002	27.83 %	27.07 %
8	(individual)	1.8219	1.7449	1.5052	17.38 %	13.73 %
12	(individual)	1.7363	1.554	0.8386	51.70 %	13.73 %
4	(1 array)	3.1171	3.0155	2.7059	51.72 %	46.04 %
8	(2 arrays)	2.8733	2.5398	2.2045	13.19 %	10.27 %
12	(3 arrays)	2.6626	2.3998	1.9398	27.15 %	19.17 %

El algoritmo propuesto mejora mucho más el resultado obtenido de forma manual para micrófonos independientes que para arrays de micrófonos debido a la menor libertad de movimiento de los sensores dentro de las salas.

4.4.3.2 Comprobación de convergencia

En la Figura 4.19 se muestra la evolución del resultado del algoritmo genético propuesto para 12 micrófonos (o 3 arrays) para todas las combinaciones de salas, para analizar si los parámetros óptimos del algoritmo genético establecidos en 4.4.2 son válidos.

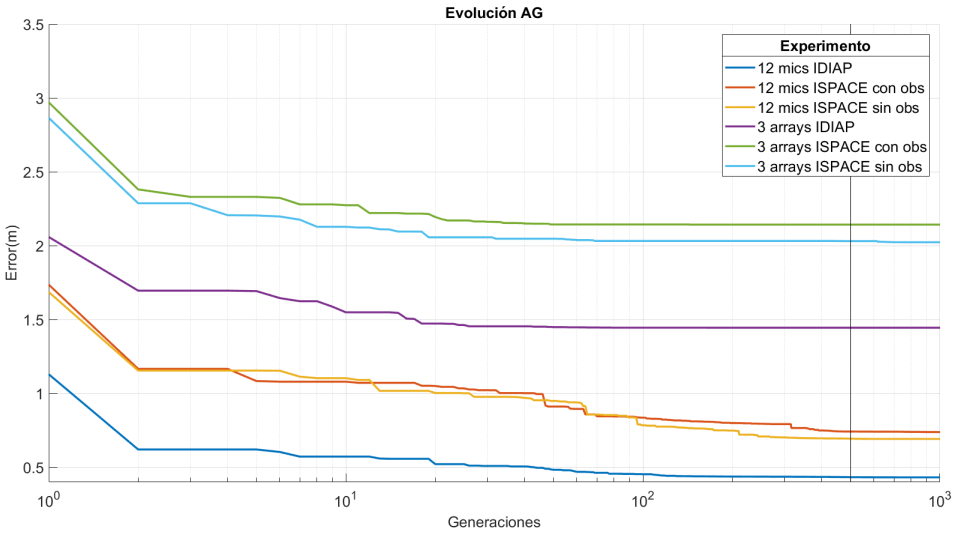


Figura 4.19: Evolución resultado algoritmo de posicionamiento para 12 micrófonos (o 3 arrays)

Como se puede observar para todos los casos se alcanza la convergencia para un número de generaciones menor que 500 como sucedía para el caso de estudio propuesto para este análisis en 4.4.2.

Por tanto, se puede afirmar que los parámetros óptimos seleccionados cumplen los requisitos requeridos.

4.4.3.3 Estudio multi-camino

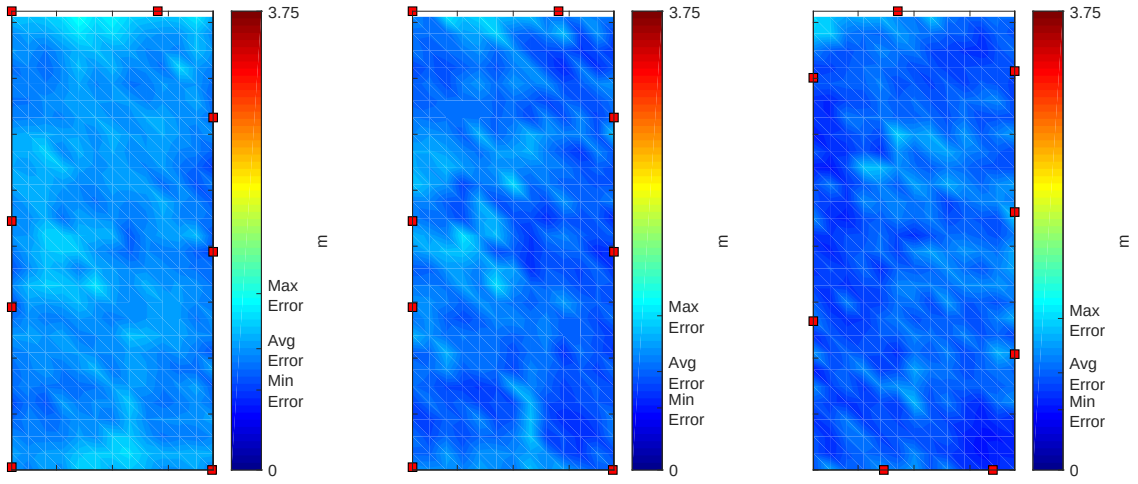
El sistema desarrollado en este trabajo es capaz de simular también el modelo acústico multi-camino, es decir, teniendo en cuenta los rebotes de las ondas acústicas.

En la Figura 4.20 muestra una comparativa del posicionamiento óptimo de micrófonos con y sin rebotes para 8 micrófonos aislados en la sala IDIAP.

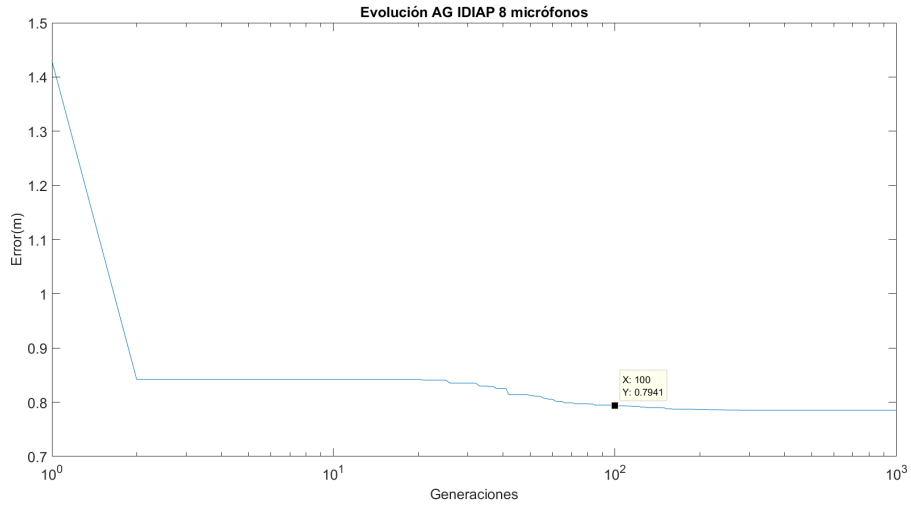
Debido al aumento sustancial del tiempo de procesamiento por generación al simular el modelo acústico con 1 rebote (70 minutos frente a 2 minutos para una población de 120 individuos) producido por el aumento de puntos a analizar que aparecen al calcular los puntos imagen, en la Figura 4.20 se realiza la comparativa para 100 generaciones.

La Figura 4.20a muestra la solución del algoritmo propuesto teniendo en cuenta 1 rebote, en la Figura 4.20b aparece el mismo posicionamiento óptimo pero calculando el error medio sin tener en cuenta rebotes y en la Figura 4.20c se representa la el posicionamiento óptimo obtenido para el mismo número de generaciones sin tener en cuenta el rebote. Por último la gráfica de la Figura 4.20d muestra

la evolución del experimento de la Figura 4.20c marcando claramente el objetivo que se alcanza para 100 generaciones.



(a) Óptimo 1 rebote (b) Sin rebote equivalente (c) Óptimo sin rebote



(d) Evolución algoritmo sin rebotes.

Figura 4.20: Comparativa del algoritmo propuesto para 0 y 1 rebote.

Comparando el error medio, de 0.8044 metros, obtenido en la Figura 4.20b correspondiente al error medio equivalente sin rebotes, con el error medio, de 0.7941 metros, obtenido en la Figura 4.20c, se puede afirmar que para el problema de estudio y para el mismo número de generaciones se obtiene un resultado similar en valor medio de error.

Teniendo en cuenta que el cálculo de un rebote no afecta de forma significativa a los resultados, y provoca un aumento muy considerable en el tiempo de procesamiento, se ha decidido realizar el resto de experimentos considerando únicamente el rayo directo.

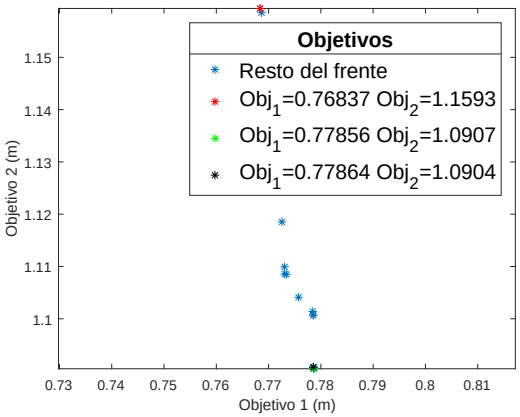
4.4.4 Simulaciones multi-objetivo

Para el análisis multi-objetivo se optado por una comparación para la configuración de 8 micrófonos aislados, para todas las salas, ejecutando 100 generaciones.

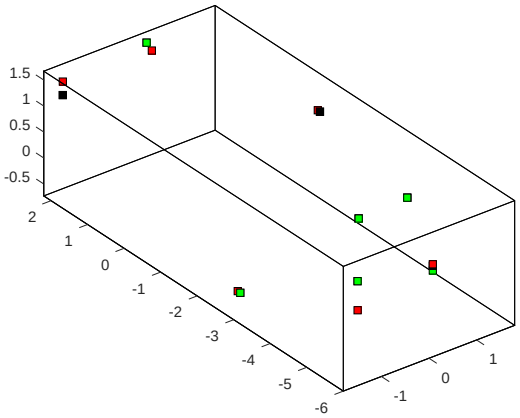
En cuanto a los objetivos a simular, se han tomado por parejas las 3 combinaciones posibles con los 3 objetivos considerados en este trabajo: el error medio (Avg Error), el error máximo (Max Error) y la diferencia entre el error máximo y el error mínimo(Max Error-Min Error).

En cuanto al criterio de selección de mejor solución se ha dejado libertad para generar el frente de pareto de soluciones que minimizan ambos objetivos, dando la posibilidad posterior de analizar los distintos objetivos de las soluciones para decidir cual es la mejor solución.

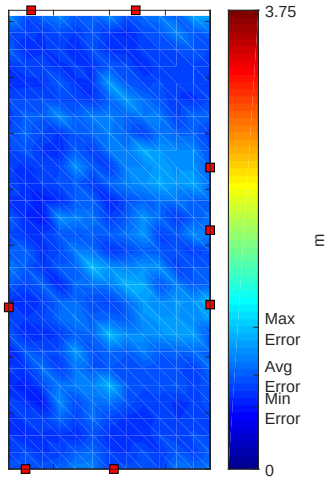
En la Figura 4.21 se muestra la solución multi-objetivo para la sala IDIAP para optimizar el error medio y el error máximo. Como se puede observar en la Figura 4.21a no existe una única solución optima para el problema, sino que aparece un frente de pareto de soluciones, de las cuales en la Figura 4.21b es muestran representadas las mas representativas en el espacio tridimensional.



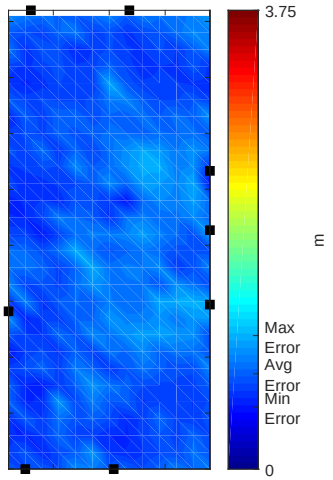
(a) Frente de pareto de soluciones.



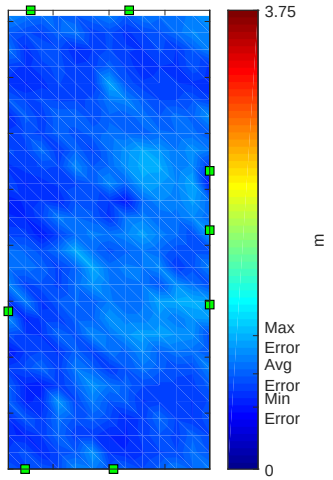
(b) Frente de pareto de soluciones.



(c) Min Objetivo 1



(d) Objetivos mínimos

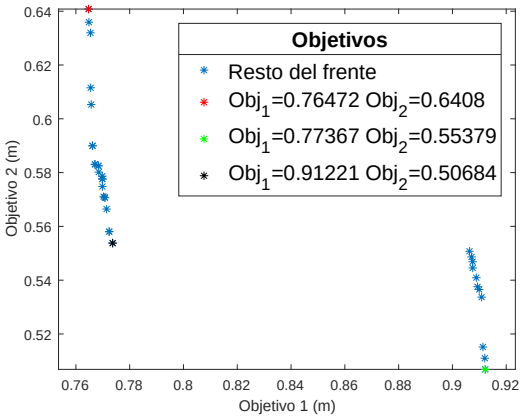


(e) Min Objetivo 2

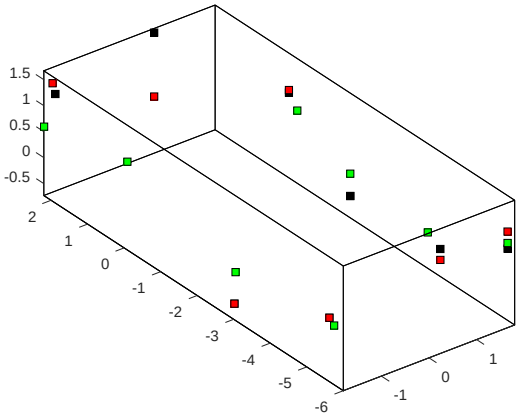
Figura 4.21: Sala IDIAP: Comparativa objetivo error medio y error máximo.

De la Figura 4.21c a la 4.21e se representan las tres soluciones más relevantes del pareto, que son la que minimiza el primer objetivo (4.21c en este caso el error medio), la que minimiza el segundo objetivo (4.21e en este caso el error máximo) y la que minimiza ambos objetivos (4.21d).

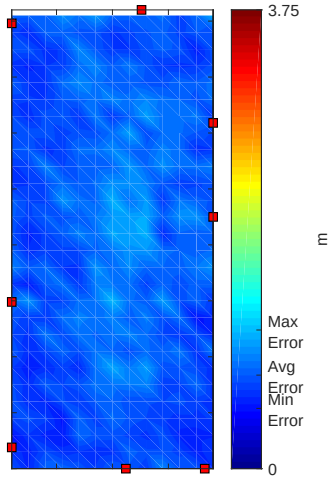
Como el caso anterior en la Figura 4.22 se muestra la solución multi-objetivo para la sala IDIAP para optimizar en este caso el error medio y la diferencia entre el error medio y el mínimo. En la Figura 4.22a aparece igualmente un frente de pareto de soluciones en función del criterio a optimizar, de las cuales en la Figura 4.22b es muestran representadas las mas representativas en el espacio tridimensional.



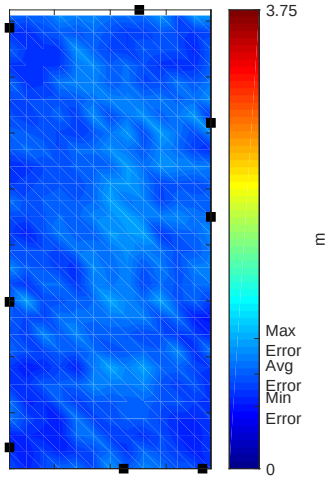
(a) Frente de pareto de soluciones.



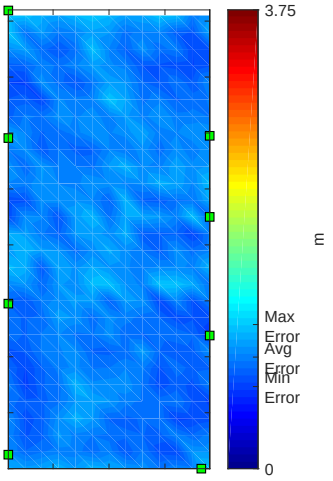
(b) Frente de pareto de soluciones.



(c) Min Objetivo 1



(d) Objetivos mínimos



(e) Min Objetivo 2

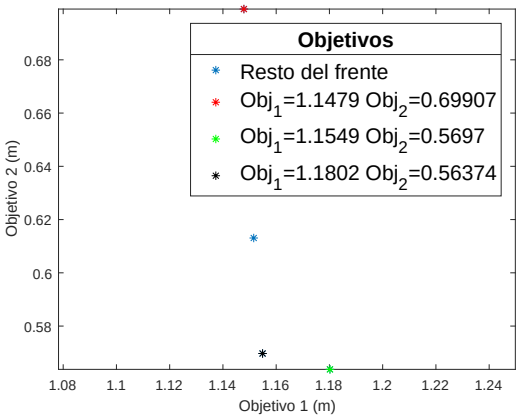
Figura 4.22: Sala IDIAP: Comparativa objetivo error medio y desviación entre error medio y mínimo.

De la Figura 4.22c a la 4.22e se representan las tres soluciones más relevantes del pareto, que son la que minimiza el primer objetivo (4.22c en este caso el error medio), la que minimiza el segundo objetivo (4.22e en este caso el la diferencia entre el error máximo y el mínimo) y la que minimiza ambos objetivos (4.22d).

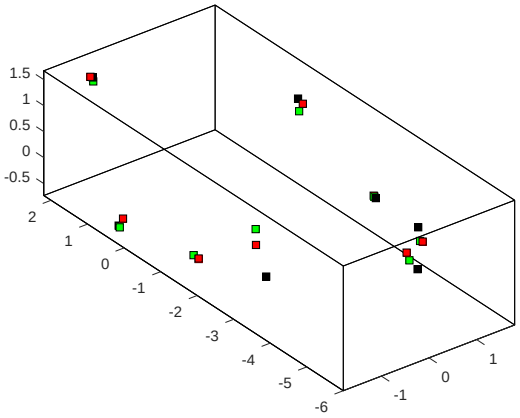
Como los casos anteriores la Figura 4.23 muestra la solución multi-objetivo para la sala IDIAP para optimizar en este caso el error máximo y la diferencia entre el error máximo y el mínimo. Obteniendo como solución el frente de pareto de soluciones de la Figura 4.23a en función del criterio a optimizar, de las cuales en la Figura 4.23b es muestran representadas las mas representativas en el espacio tridimensional.

De la Figura 4.23c a la 4.23e se representan las tres soluciones más relevantes del pareto, que son la que minimiza el primer objetivo (4.23c en este caso el error máximo), la que minimiza el segundo objetivo (4.23e en este caso el la diferencia entre el error máximo y el mínimo) y la que minimiza ambos objetivos (4.23d).

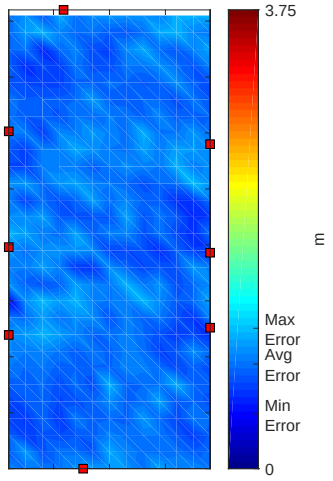
De los resultados anteriores se puede ver como para la sala IDIAP debido a la simetría de la sala y a la dependencia parcial entre los objetivos se obtienen resultados que visualmente similares (en las proyecciones bidimensionales), aunque algunas soluciones cambian considerablemente en el espacio tridimensional.



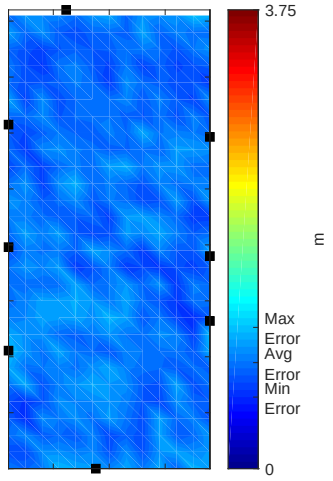
(a) Frente de pareto de soluciones.



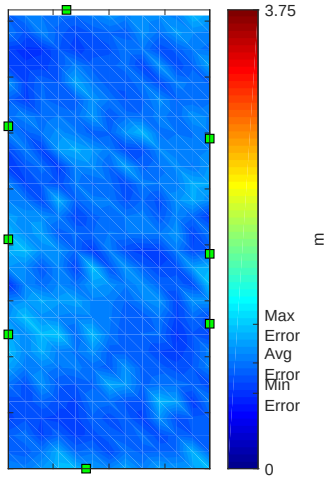
(b) Frente de pareto de soluciones.



(c) Min Objetivo 1



(d) Objetivos mínimos



(e) Min Objetivo 2

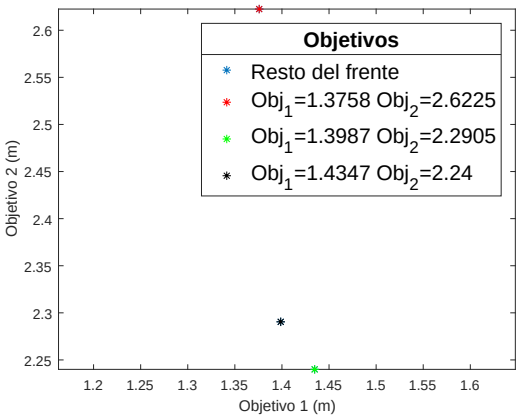
Figura 4.23: Sala IDIAP: Comparativa objetivo error máximo y desviación entre error máximo y mínimo.

Además se puede observar como la simulación que genera un frente de pareto con mayor variación es aquella que optimiza el error medio y la diferencia entre el máximo y el mínimo, por lo que se puede afirmar que para esta sala es la que genera un conjunto de soluciones mejor, ya que genera un conjunto más disperso de soluciones, con mayor separación entre los valores máximos y mínimos para ambos objetivos, que es el objetivo principal del análisis multi-objetivo.

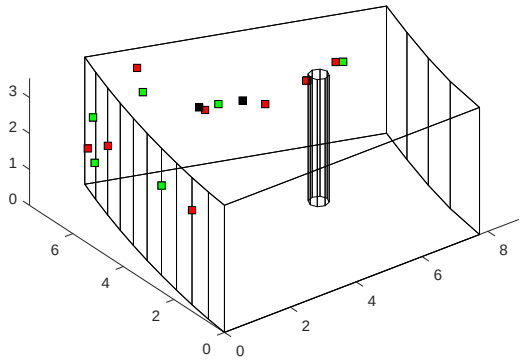
La Figura 4.24 muestra la solución multi-objetivo para la sala ISPACE sin rebotes para optimizar en este caso el error medio y el error máximo. Obteniendo como solución el frente de pareto de soluciones de la Figura 4.24a en función del criterio a optimizar, de las cuales en la Figura 4.24b es muestran representadas las mas representativas en el espacio tridimensional.

De la Figura 4.24c a la 4.24e se representan las tres soluciones más relevantes del pareto, que son la que minimiza el primer objetivo (4.24c en este caso el error medio), la que minimiza el segundo objetivo (4.24e en este caso el error máximo) y la que minimiza ambos objetivos (4.24d).

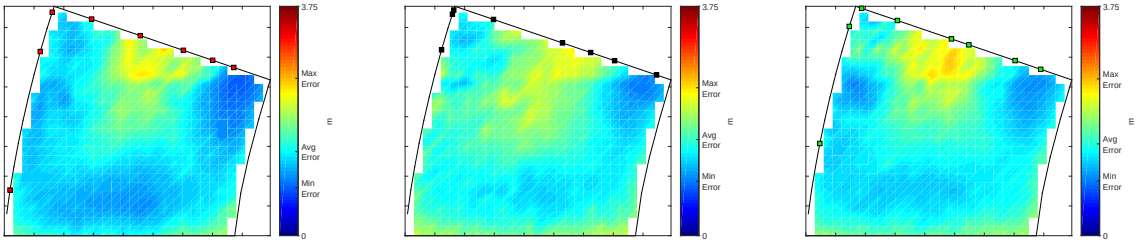
Como en el caso anterior la Figura 4.25 muestra la solución multi-objetivo para la sala ISPACE sin rebotes para optimizar en este caso el error medio y la diferencia entre el error máximo y el error mínimo. Obteniendo como solución el frente de pareto de soluciones de la Figura 4.25a en función del criterio a optimizar, de las cuales en la Figura 4.25b es muestran representadas las mas representativas en el espacio tridimensional.



(a) Frente de pareto de soluciones.

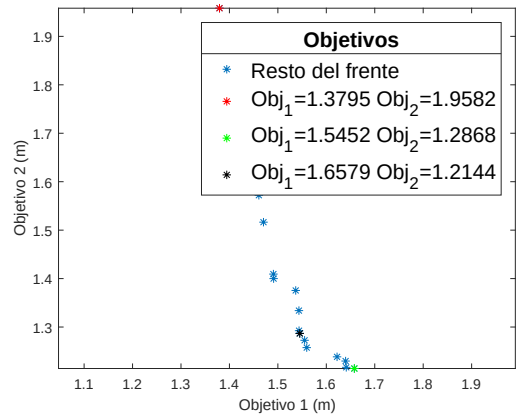


(b) Frente de pareto de soluciones.

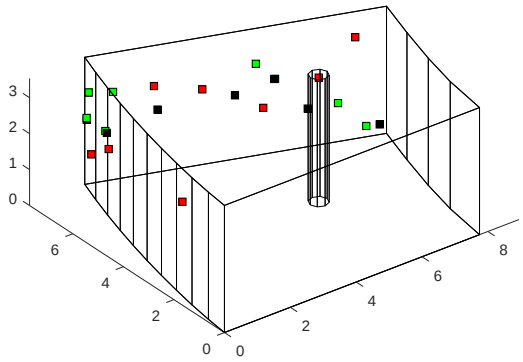


(c) Min Objetivo 1 (d) Objetivos mínimos (e) Min Objetivo 2

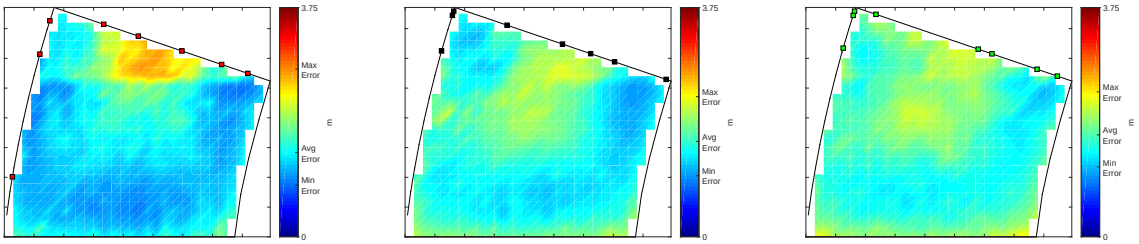
Figura 4.24: Sala ISPACE sin obstáculos: Comparativa objetivo error medio y error máximo.



(a) Frente de pareto de soluciones.



(b) Frente de pareto de soluciones.



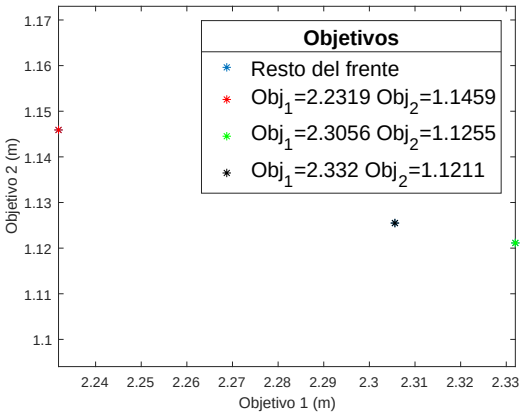
(c) Min Objetivo 1 (d) Objetivos mínimos (e) Min Objetivo 2

Figura 4.25: Sala ISPACE sin obstáculos: Comparativa objetivo error medio y desviación entre error máximo y mínimo.

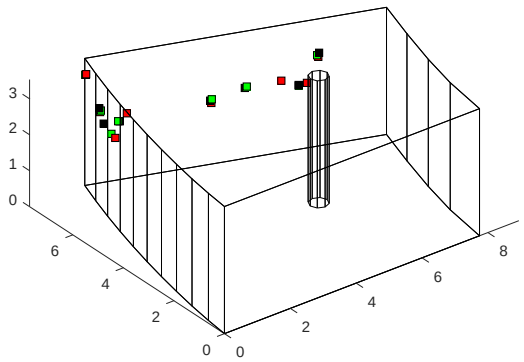
De la Figura 4.25c a la 4.25e se representan las tres soluciones más relevantes del pareto, que son la que minimiza el primer objetivo (4.25c en este caso el error medio), la que minimiza el segundo objetivo

(4.25e en este caso la diferencia entre el error máximo y el error mínimo) y la que minimiza ambos objetivos (4.25d).

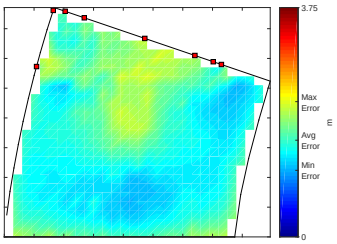
Como los casos anteriores en la Figura 4.26 se muestra la solución multi-objetivo para la sala ISPACE sin rebotes para optimizar en este caso el error máximo y la diferencia entre el error máximo y el error mínimo. Obteniendo como solución el frente de pareto de soluciones de la Figura 4.26a en función del criterio a optimizar, de las cuales en la Figura 4.26b es muestran representadas las mas representativas en el espacio tridimensional.



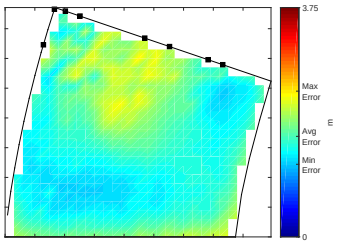
(a) Frente de pareto de soluciones.



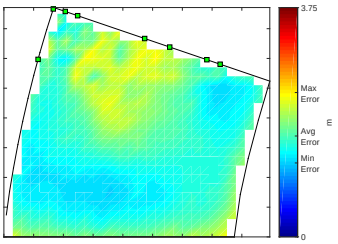
(b) Frente de pareto de soluciones.



(c) Min Objetivo 1



(d) Objetivos mínimos



(e) Min Objetivo 2

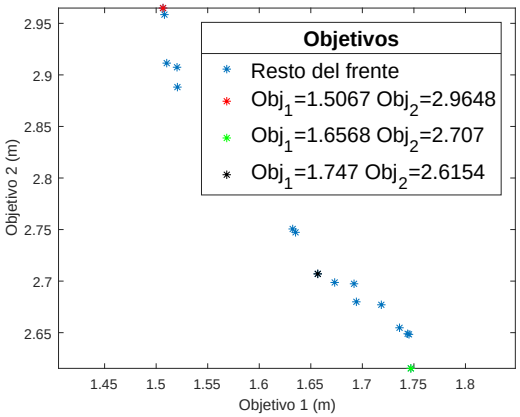
Figura 4.26: Sala ISPACE sin obstáculos: Comparativa objetivo error máximo y desviación entre error máximo y mínimo.

De la Figura 4.26c a la 4.26e se representan las tres soluciones más relevantes del pareto, que son la que minimiza el primer objetivo (4.26c en este caso el error máximo), la que minimiza el segundo objetivo (4.26e en este caso la diferencia entre el error máximo y el error mínimo) y la que minimiza ambos objetivos (4.26d).

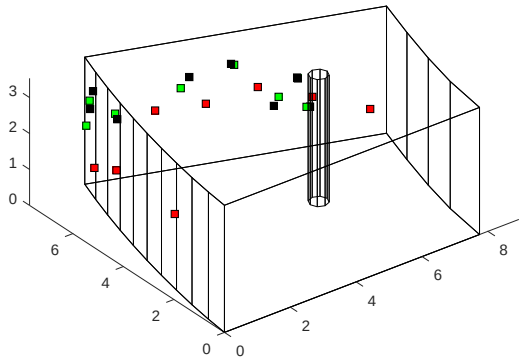
Para el caso de la sala ISPACE sin obstáculos al tener mayores dimensiones y menos simetría, hay mayor libertad de posicionamiento, por lo que existe más variación entre los resultados dependiendo del objetivo a priorizar. Al igual que para el caso de IDIAP las combinaciones donde se tiene en cuenta como objetivo el error medio hay más diferencia entre las soluciones con distintos criterios.

La Figura 4.27 muestra la solución multi-objetivo para la sala ISPACE con rebotes para optimizar en este caso el error medio y el error máximo. Obteniendo como solución el frente de pareto de soluciones de la Figura 4.27a en función del criterio a optimizar, de las cuales en la Figura 4.27b es muestran representadas las mas representativas en el espacio tridimensional.

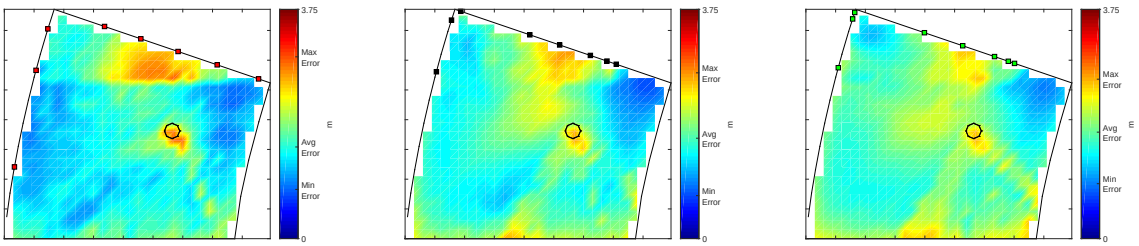
De la Figura 4.27c a la 4.27e se representan las tres soluciones más relevantes del pareto, que son la que minimiza el primer objetivo (4.27c en este caso el error medio), la que minimiza el segundo objetivo (4.27e en este caso el error máximo) y la que minimiza ambos objetivos (4.27d).



(a) Frente de pareto de soluciones.



(b) Frente de pareto de soluciones.



(c) Min Objetivo 1

(d) Objetivos mínimos

(e) Min Objetivo 2

Figura 4.27: Sala ISPACE con obstáculos: Comparativa objetivo error medio y error máximo.

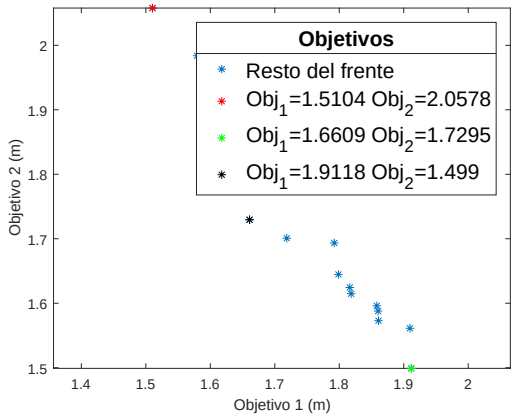
Como en el caso anterior la Figura 4.28 muestra la solución multi-objetivo para la sala ISPACE con rebotes para optimizar en este caso el error medio y la diferencia entre el error máximo y el error mínimo. Obteniendo como solución el frente de pareto de soluciones de la Figura 4.28a en función del criterio a optimizar, de las cuales en la Figura 4.28b es muestran representadas las mas representativas en el espacio tridimensional.

De la Figura 4.28c a la 4.28e se representan las tres soluciones más relevantes del pareto, que son la que minimiza el primer objetivo (4.28c en este caso el error medio), la que minimiza el segundo objetivo (4.28e en este caso el la diferencia entre el error máximo y el error mínimo) y la que minimiza ambos objetivos (4.28d).

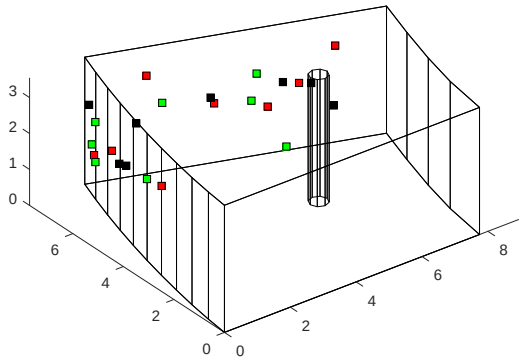
Como los casos anteriores la Figura 4.29 representa la solución multi-objetivo para la sala ISPACE con rebotes para optimizar en este caso el error máximo y la diferencia entre el error máximo y el error mínimo. Obteniendo como solución el frente de pareto de soluciones de la Figura 4.29a en función del criterio a optimizar, de las cuales en la Figura 4.29b es muestran representadas las mas representativas en el espacio tridimensional.

De la Figura 4.29c a la 4.29e se representan las tres soluciones más relevantes del pareto, que son la que minimiza el primer objetivo (4.29c en este caso el error máximo), la que minimiza el segundo objetivo (4.29e en este caso el la diferencia entre el error máximo y el error mínimo) y la que minimiza ambos objetivos (4.29d).

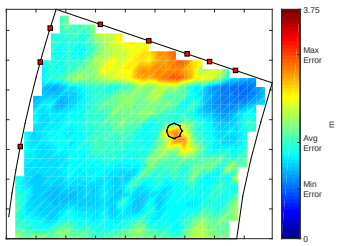
Para el caso de la sala ISPACE con obstáculos al tener mayores dimensiones, obstáculos y menos simetría, hay mayor libertad de posicionamiento, por lo que existe más variación entre los resultados dependiendo del objetivo a priorizar. Al igual que para los otros dos casos las combinaciones donde se tiene en cuenta como objetivo el error medio hay más diferencia entre las soluciones con distintos criterios, ya que tiene menos dependencia con respecto a los otros dos objetivos que la otra combinación.



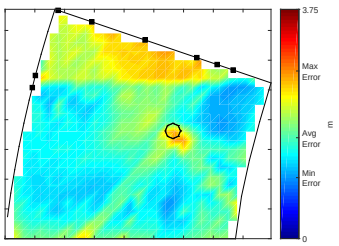
(a) Frente de pareto de soluciones.



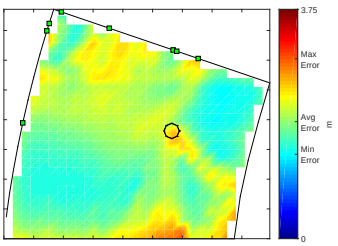
(b) Frente de pareto de soluciones.



(c) Min Objetivo 1

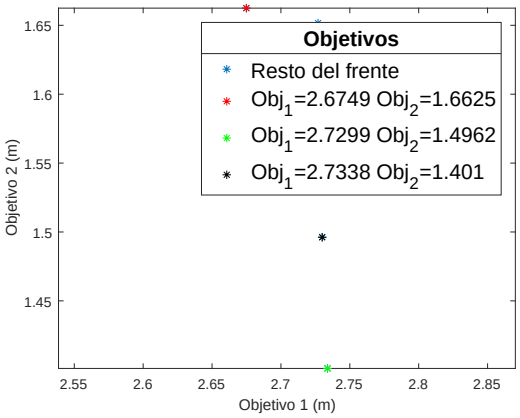


(d) Objetivos mínimos

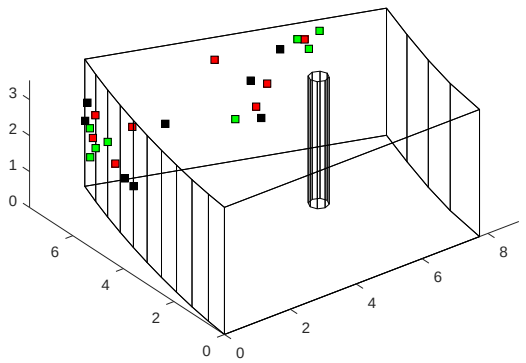


(e) Min Objetivo 2

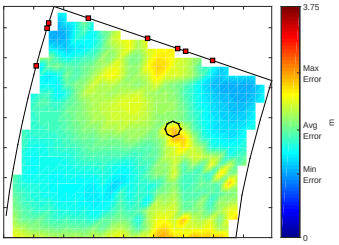
Figura 4.28: Sala ISPACE con obstáculos: Comparativa objetivo error medio y desviación entre error máximo y mínimo.



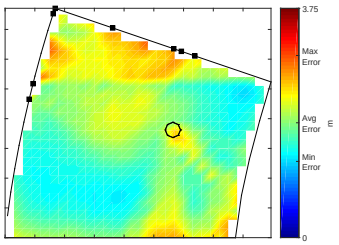
(a) Frente de pareto de soluciones.



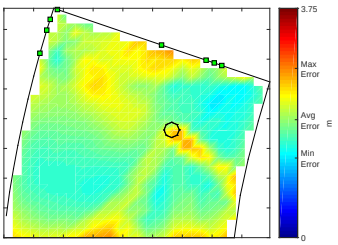
(b) Frente de pareto de soluciones.



(c) Min Objetivo 1



(d) Objetivos mínimos



(e) Min Objetivo 2

Figura 4.29: Sala ISPACE con obstáculos: Comparativa objetivo error máximo y desviación entre error máximo y mínimo.

En las Tablas 4.4, 4.5 y 4.6 se hace un análisis cuantitativo de los distintas combinaciones de objetivos de forma análoga a como se hizo para el algoritmo multi-objetivo. Donde en la columna *Opt* aparecen los mejores resultados obtenidos para cada objetivo de la combinación, comparándolos respecto a los obtenidos con un posicionamiento aleatorio (columna *Rnd*), y con respecto al posicionamiento manual (columna *Man*). En las dos últimas columnas, se presentan también las mejoras relativas del posicionamiento óptimo en comparación con el posicionamiento aleatorio ($\Delta_p^r = (Rnd - Opt)/Rnd$) y el posicionamiento manual ($\Delta_p^m = (Man - Opt)/Man$).

Tabla 4.4: Comparación del posicionamiento óptimo multi-objetivo teniendo en cuenta el error medio y el error máximo.

Sala	*Rnd*		*Man*		*Opt*		Δ_p^r		Δ_p^m	
	Obj_1	Obj_2	Obj_1	Obj_2	Obj_1	Obj_2	Obj_1	Obj_2	Obj_1	Obj_2
IDIAP	0.8319	1.1964	0.9652	2.0059	0.7684	1.0904	7.63 %	8.85 %	20.39 %	45.64 %
ISPACE sin obst	1.7574	3.7999	1.6369	4.4669	1.3758	2.4000	21.71 %	41.05 %	15.95 %	48.85 %
ISPACE con obst	1.8219	3.7999	1.7449	4.4669	1.5067	2.6154	17.30 %	31.17 %	13.65 %	41.45 %

Tabla 4.5: Comparación del posicionamiento óptimo multi-objetivo teniendo en cuenta el error medio y la diferencia entre el error máximo y el error mínimo.

Sala	*Rnd*		*Man*		*Opt*		Δ_p^r		Δ_p^m	
	Obj_1	Obj_2	Obj_1	Obj_2	Obj_1	Obj_2	Obj_1	Obj_2	Obj_1	Obj_2
IDIAP	0.8301	0.6842	0.9652	1.4460	0.7648	0.5068	43.20 %	63.49 %	20.76 %	64.95 %
ISPACE sin obst	1.7574	3.1000	1.6369	3.6577	1.3795	1.2144	21.50 %	60.83 %	15.72 %	66.80 %
ISPACE con obst	1.1822	3.1000	1.7449	3.6075	1.5104	1.4999	16.65 %	51.62 %	13.44 %	58.42 %

Tabla 4.6: Comparación del posicionamiento óptimo multi-objetivo teniendo en cuenta el error máximo y la diferencia entre el error máximo y el error mínimo.

Sala	*Rnd*		*Man*		*Opt*		Δ_p^r		Δ_p^m	
	Obj_1	Obj_2	Obj_1	Obj_2	Obj_1	Obj_2	Obj_1	Obj_2	Obj_1	Obj_2
IDIAP	1.2513	0.6229	2.0059	1.4460	1.1479	0.5637	48.36 %	59.39 %	42.77 %	61.02 %
ISPACE sin obst	3.7999	3.1000	4.4669	3.6577	2.2319	1.1211	41.26 %	63.84 %	50.03 %	69.35 %
ISPACE con obst	3.7999	3.1	4.4669	3.6075	2.6749	1.4010	29.60 %	54.81 %	40.12 %	61.16 %

Si se comparan cada objetivo de cada combinación que son iguales entre sí por cada sala, p.e. el error medio obtenido con la combinación del error medio y del error máximo y el error medio obtenido con la combinación del error medio y de la diferencia de error máximo y error mínimo para la sala IDIAP, se obtienen valores dentro del mismo rango para cada cada objetivo y cada sala, por lo que se demuestra que el algoritmo genético multi-objetivo funciona como se espera.

Además, si se comparan los objetivos de error medio para las combinaciones de sala y objetivos en las que interviene, con los resultados obtenidos con el algoritmo multi-objetivo, se puede ver como el error medio obtenido se encuentra dentro del mismo rango, por lo que se puede asumir que ambos algoritmos funcionan correctamente para un único objetivo.

Capítulo 5

Conclusiones y líneas futuras

5.1 Introducción

En este capítulo se abordarán las conclusiones alcanzadas durante la realización de este trabajo y tras su finalización y se propondrá un conjunto de líneas de trabajo futuras que han ido apareciendo durante la realización del mismo.

5.2 Conclusiones

El objetivo principal que se ha perseguido en este trabajo es el desarrollo de un *Sistema de posicionamiento óptimo de micrófonos en entornos cerrados basados en optimización multi-objetivo para tareas de localización*. Además como este trabajo como parte de la línea de investigación a la que pertenece este trabajo, iniciada en [15] y [16], al objetivo principal de este trabajo se añaden los objetivos planteados como líneas futuras de los trabajos anteriores.

Para la consecución de este objetivo se ha partido del estudio teórico en profundidad, apoyado de la revisión del estado del arte, de las distintas técnicas de posicionamiento óptimo multi-objetivo de sensores en entornos cerrados y de los modelos de propagación acústica utilizables para la obtención del error de localización.

Este estudio teórico aborda en primer lugar el problema de la localización, para obtener el error de localización necesario para la optimización. Para ello se estudian los distintos métodos de localización en espacios cerrados, ya sean basados en técnicas basadas en la información recibida por los sensores para todo tipo de sensores (de la que destaca el método de localización basado en TDOA), como las técnicas específicas para la localización de micrófonos basadas en *beamforming* (entre las que destaca el método de localización basado en SRP). Posteriormente se ha estudiado los distintos modelos acústicos posibles, de los cuales se profundiza en el modelo de fuente imagen, basado en teoría geométrica. Este método es necesario para la obtención del multicamino para el modelo basado en SRP-PHAT utilizado para obtener el patrón SRP para cada punto de la sala, utilizado para calcular la SLF, que permite conocer la probabilidad con la que el objetivo puede encontrarse en cada punto del espacio, necesaria para el cálculo del error de localización.

Una vez finalizado el estudio del problema de localización se pasa al estudio del problema de posicionamiento óptimo de sensores. Además del estudio de los algoritmos genéticos multi-objetivo, como parte de la línea de investigación este trabajo también aborda la optimización mono-objetivo (ya que en los

trabajos anteriores, se observó que el uso de las librerías definidas en Matlab, no funcionan de forma correcta para el problema que se aborda en este trabajo) por lo que se ha realizado también un estudio en profundidad de los algoritmos genéticos mono-objetivo.

Para la consecución del objetivo principal del desarrollo del sistema de posicionamiento óptimo se ha sub-dividido el problema en los siguientes tres grandes bloques.

- La obtención del error de localización: Este sub-sistema calcula el error para cada punto de la sala (en una rejilla bidimensional a una altura fija de 1.7m) calculando el error mediante el modelo basado en SRP utilizado en este trabajo, adaptado del desarrollado por Jose Velasco en [17], apoyado por el modelo acústico de fuente-imagen desarrollado en este trabajo. El coste computacional de este modelo acústico es bastante grande, por lo que para solucionarlo se ha dividido el proceso de la obtención de los puntos imagen y su posterior validación de esos puntos en 2. La primera parte de este proceso obtiene los puntos imagen de forma matemática obteniendo los puntos simétricos para cada punto de la sala con respecto a cada superficie, y posteriormente calcula para cada punto de la sala y los puntos imagen las zonas ciegas de las superficies que tiene cada punto. La segunda parte, que se realiza para el cálculo de cada solución, se encarga de verificar si las posiciones de los micrófonos se encuentra en alguna de las zonas ciegas de cada punto de la sala obtenidas previamente para discretizar si ese punto esta cegado para los alguno de los micrófonos de la solución, solucionando el tratamiento de obstáculos en las salas.

- La optimización del error de localización: Este sub-sistema permite la optimización mono-objetivo y multi-objetivo del error de localización. Los objetivos a minimizar en este trabajo se calculan en función del error de localización siendo el error medio para cada punto de la sala, el error máximo que se obtiene para los puntos y la diferencia entre el error máximo y el error mínimo, para minimizar la varianza del error. Este sub-sistema ha sido realizado con la mayor variedad de funciones de mutación y cruce posibles, añadiendo la posibilidad de corrección para mejorar el funcionamiento y reduciendo el coste computacional de recalcular los objetivos para la ultima población guardando los datos previamente, ya que el tiempo obtenido en el cálculo del objetivo es considerable.

- Sistema global: es el sistema que engloba ambas partes, diseñado con una interfaz gráfica para la facilidad de su manejo.

Por ultimo se ha realizado una evaluación exhaustiva y rigurosa del sistema desarrollado, comprobando el funcionamiento del sistema para salas tanto teóricas como reales, de distintos tamaños y formas, para un conjunto grande de condiciones experimentales.

En los resultados mostrados, se aprecian varios tipos de errores con distintos rangos. A continuación se detalla el porqué de cada uno de ellos:

- En localización basada en TDOA está representada la solución de la CRLB, que es dependiente de la distancia al cuadrado por lo que cuanto más grande es la sala mucho mayor es este valor. Este no lleva valores unitarios ya que representa un error relativo.

- En la localización basada en SRP está representado el error medio para cada punto. Se representa en metros.

En cuanto al tiempo de procesamiento para la obtención de la solución óptima, éste es muy dependiente del tiempo de proceso para cada solución, siendo mucho mayor el tiempo de procesamiento para la localización basada en SRP y siendo el tiempo de procesamiento de ésta muy dependiente del número de rebotes utilizado para el cálculo de la solución óptima.

Se ha demostrado empíricamente cuales son los parámetros óptimos del algoritmo para el problema tratado, calculado previamente y corroborado analizando la convergencia posteriormente después de obtener los resultado.

A través del análisis cuantitativo del error medio de localización mono-objetivo, se ha demostrado que el algoritmo desarrollado mejora sustancialmente la solución comparándolo con un posicionamiento aleatorio y un posicionamiento manual.

Con el análisis cuantitativo del error de localización multi-camino comparado con el error mono-camino se ha demostrado que para este problema, teniendo en cuenta el aumento muy significativo del coste computacional. En una primera etapa sólo sería necesario utilizar el posicionamiento utilizando la localización mono-camino, aunque para la utilización en un caso real es recomendable realizar posteriormente una segunda etapa de posicionamiento utilizando el cálculo del multi-camino.

Se ha demostrado con el análisis cuantitativo de los objetivos del algoritmo genético, tanto mono-objetivo como multi-objetivo, como el algoritmo funciona correctamente para todas las salas que se han probado.

En conclusión se han conseguido todos los objetivos planteados de este trabajo, así como los objetivos planteados como líneas futuras de los trabajos anteriores.

5.3 Líneas futuras

A continuación se describen las futuras futuras mejoras que se han ido planteando durante la realización de este proyecto.

5.3.1 Estudio multi-camino exhaustivo

Aunque en este trabajo se ha comprobado que no existen diferencias considerables entre hacer un análisis multi-camino del error o no, podría plantearse realizar este estudio para salas con muchos obstáculos o de formas irregulares.

5.3.2 Error en todo el espacio

En este trabajo se hace un análisis del error para una rejilla a una altura fija, en una versión futura el error a optimizar y mostrar podría calcularse para todo el espacio de la sala.

5.3.3 Añadir nuevos tipos de sensores y topologías de arrays de micrófonos

Además de los sensores con localización basada en TDOA y los micrófonos usados en este trabajo podrían introducirse nuevos tipos de sensores con sus respectivas funciones de error de localización y las restricciones de ubicación que fueran necesarias (para algunas topologías de arrays de micrófonos es útil posicionarlos dentro del espacio de la sala).

5.3.4 Comparativa con otros algoritmos metaheurísticos

Para corroborar el funcionamiento correcto del algoritmo genético para este tipo de problemas es recomendable compararlo con otro tipo de algoritmos metaheurísticos.

Capítulo 6

Pliego de condiciones

Para la correcta utilización del sistema desarrollado en este trabajo, debe disponerse de un hardware y un software que cumpla unos requisitos mínimos.

6.1 Requisitos de Hardware

- Procesador multinúcleo 32/64 bits 2GHz o superior, recomendado al menos 8 núcleos.
- 1 GB de memoria RAM o superior, recomendado al menos 8 GB.
- Recomendado el uso de pantalla panorámica para la visualización de la interfaz gráfica.
- Interfaz de red para enviar las simulaciones al cluster.
- Al menos 150 MB libres en el disco duro para las funciones y datos, recomendado disponer de al menos 1 GB para almacenar los resultados.

6.2 Requisitos de Software

- Windows 7 o superior.
- Matlab 2012a o superior, recomendado utilizar la versión 2017a.

Capítulo 7

Presupuesto

7.1 Costes de equipamiento

- Equipamiento hardware utilizados:

Tabla 7.1: Costes de equipamiento hardware

Concepto	**Cantidad**	**Coste Unitario**	**Subtotal(€)**
PC Intel Core I5	1	500 €	500 €
Coste total HW			**500 €**

- Recursos software utilizados:

Tabla 7.2: Costes de recursos software

Concepto	**Cantidad**	**Coste Unitario**	**Subtotal (€)**
Windows 8	1	500 €	500 €
Matlab 2017	1	2000 €	2000 €
Software LaTeX	1	0 €	0 €
Coste total SW			**2500 €**

7.2 Costes de mano de obra

Tabla 7.3: Costes debidos a mano de obra

Concepto	**Cantidad**	**Coste Unitario**	**Subtotal (€)**
Desarrollo SW	300	65 €/hora	19500 €
Mecanografiado y maquetado de este documento	100	15 €/hora	1500 €
Coste total HW			**21000 €**

El número de horas de ingeniería equivale a cuatro meses a media jornada. El mecanografiado y maquetado se ha estimado en 5 semanas a media jornada.

7.3 Coste total del Presupuesto

Tabla 7.4: Coste total del presupuesto

Concepto	**Subtotal (€)**
Costes de equipamiento hardware	500 €
Costes de recursos software	2500 €
Costes mano de obra	21000 €
Coste total HW	**24000 €**

El importe total del proyecto asciende a la cantidad de: *VENTICUATRO MIL EUROS*

En Alcalá de Henares,___ de _____ de 20___.

Fdo: Roberto Macho Pedroso

Ingeniero de Telecomunicación

Bibliografía

[1] R. Pérez-Pascual, "Diseño, implementación y evaluación de un sistema de simulación acústica en entornos cerrados," Master's thesis, Universidad de Alcalá. Escuela Politécnica Superior, 2016.

[2] C. A. Correa Flórez, R. A. Bolaños, and A. Molina Cabrera, "Algoritmo multiobjetivo nsga-ii aplicado al problema de la mochila." *Scientia et technica*, vol. 14, no. 39, 2008.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[4] V. H. G. Antonio José Álvarez Gamero, "Optimización de la distribución en planta de instalaciones industriales mediante algoritmos genéticos," Master's thesis, Universidad de Sevilla, 2010.

[5] S. A. Zekavat and R. M. Buehrer, *Handbook of Position Location: Theory, Practice, and Advances*. IEEE, 2012.

[6] A. N. Bishop, B. Fidan, B. Anderson, K. Doğançay, and P. N. Pathirana, "Optimality analysis of sensor-target localization geometries," *Automatica*, vol. 46, no. 3, pp. 479–492, 2010.

[7] C. Yang, L. Kaplan, and E. Blasch, "Performance measures of covariance and information matrices in resource management for target state estimation," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 48, no. 3, pp. 2594–2613, Jul. 2012.

[8] D. Hong, J. Ortiz, K. Whitehouse, and D. E. Culler, "Towards automatic spatial verification of sensor placement in buildings," in *BuildSys 2013, Proceedings of the 5th ACM Workshop On Embedded Systems For Energy-Efficient Buildings, Roma, Italy, November 13-14, 2013*, 2013, pp. 13:1–13:8.

[9] A. van den Hengel, R. Hill, B. Ward, A. Cichowski, H. Detmold, C. Madden, A. Dick, and J. Bastian, "Automatic camera placement for large scale surveillance networks," 2009, pp. 1–6.

[10] C. Cowan and P. Kovesi, "Automatic sensor placement from vision task requirements," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 3, pp. 407–416, 1988.

[11] B. Triggs and C. Laugier, "Automatic camera placement for robot vision tasks," in *Proceedings of the 1995 International Conference on Robotics and Automation, Nagoya, Aichi, Japan, May 21-27, 1995*, 1995, pp. 1732–1737.

[12] E. M. Gorostiza, J. L. Lázaro Galilea, F. J. Meca Meca, D. Salido Monzú, F. Espinosa Zapata, and L. Pallarés Puerto, "Infrared sensor system for mobile-robot positioning in intelligent spaces," *Sensors*, vol. 11, no. 5, pp. 5416–5438, 2011.

[13] M. J. Taghizadeh, S. Haghighatshoar, A. Asaei, P. N. Garner, and H. Bourlard, "Robust microphone placement for source localization from noisy distance measurements," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*, 2015, pp. 2579–2583.

[14] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT press, 1998.

[15] R. Macho-Pedroso, "Aplicación de un sistema de posicionamiento óptimo de sensores al diseño de un entorno distribuido de agrupaciones de micrófonos." Trabajo Fin de Grado. Escuela Politécnica Superior. Universidad de Alcalá, 2015.

[16] R. Macho-Pedroso, F. Domingo-Perez, J. Velasco, C. Losada, and J. Macias-Guarasa, "Optimal microphone placement for indoor acoustic localization using evolutionary optimization," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN 2016)*, 10/2016 2016, pp. 1–8.

[17] J. Velasco, C. J. Martín-Arguedas, J. Macias-Guarasa, D. Pizarro, and M. Mazo, "Proposal and validation of an analytical generative model of SRP-PHAT power maps in reverberant scenarios," *Signal Processing*, vol. 119, pp. 209 – 228, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165168415002650

[18] F. Domingo-Perez, J. L. Lazaro-Galilea, A. Wieser, E. Martin-Gorostiza, D. Salido-Monzu, and A. de la Llana, "Sensor placement determination for range-difference positioning using evolutionary multi-objective optimization," *Expert Systems with Applications*, vol. 47, pp. 95–105, 2016.

[19] J. M. Shu Wang and B. K. Yi, "Location based services for mobiles: Techonogies and stardards," 2008, http://to.swang.googlepages.com/ICC2008LBSforMobilessimplifiedR2.pdf [Último acceso 07/julio/2015].

[20] R. Kaune, "Accuracy studies for tdoa and toa localization," in *Information Fusion (FUSION), 2012 15th International Conference on*. IEEE, 2012, pp. 408–415.

[21] S.-Y. Jung, S. Hann, and C.-S. Park, "Tdoa-based optical wireless indoor localization using led ceiling lamps," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 4, 2011.

[22] J. Yang and Y. Chen, "Indoor localization using improved rss-based lateration methods," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*. IEEE, 2009, pp. 1–6.

[23] D. Liu, K. Liu, Y. Ma, and J. Yu, "Joint toa and doa localization in indoor environment using virtual stations," *IEEE Communications Letters*, vol. 18, no. 8, pp. 1423–1426, 2014.

[24] E. Martín, "Sistema de posicionamiento absoluto basado en infrarrojos para el guiado de robots móviles en espacios inteligentes." Ph.D. dissertation, Alcalá, Alcalá de Henares, 2012.

[25] F. Domingo-Perez, J. Lazaro-Galilea, E. Martin-Gorostiza, D. Salido-Monzu, and A. Wieser, "Evolutionary optimization of sensor deployment for an indoor positioning system with unknown number of anchors," in *Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS), 2014*, Nov 2014, pp. 195–202.

[26] F. Domingo, "Sensor resource management with evolutionary algorithms applied to indoor positioning," Ph.D. dissertation, UAH, 2016.

[27] S. Chepuri and G. Leus, "Sparsity-promoting sensor selection for non-linear measurement models," *Signal Processing, IEEE Transactions on*, vol. 63, no. 3, pp. 684–698, Feb 2015.

[28] J. H. DiBiase, "A high-accuracy, low-latency technique for talker localization in reverberant environments using microphone arrays," Ph.D. dissertation, Brown University, 2000.

[29] J. DiBiase, H. Silverman, and M. Brandstein, "Robust localization in reverberant rooms," *Microphone Arrays*, pp. 157–180, 2001.

[30] N. Bulusu, J. Heidemann, and D. Estrin, "Gps-less low-cost outdoor localization for very small devices," *IEEE personal communications*, vol. 7, no. 5, pp. 28–34, 2000.

[31] N. M. Drawil, H. M. Amar, and O. A. Basir, "Gps localization accuracy classification: A context-based approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 262–273, 2013.

[32] M. Hernández Hernández, "Radiobalizas y radiogoniometría: identificación y localización," 2014.

[33] H.-S. Kim and J.-S. Choi, "Advanced indoor localization using ultrasonic sensor and digital compass," in *Control, Automation and Systems, 2008. ICCAS 2008. International Conference on.* IEEE, 2008, pp. 223–226.

[34] M. Hazas and A. Hopper, "Broadband ultrasonic location systems for improved indoor positioning," *IEEE Transactions on mobile Computing*, vol. 5, no. 5, pp. 536–547, 2006.

[35] J. Cunha, E. Pedrosa, C. Cruz, A. J. Neves, and N. Lau, "Using a depth camera for indoor robot localization and navigation," *DETI/IEETA-University of Aveiro, Portugal*, 2011.

[36] R. Parhizkar, I. Dokmanic, and M. Vetterli, "Single-channel indoor microphone localization," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on.* IEEE, 2014, pp. 1434–1438.

[37] J.-M. Valin, F. Michaud, J. Rouat, and D. Létourneau, "Robust sound source localization using a microphone array on a mobile robot," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 2. IEEE, 2003, pp. 1228–1233.

[38] R. H. Lyon, R. G. DeJong, and M. Heckl, "Theory and application of statistical energy analysis," *The journal of the acoustical society of america*, vol. 98, no. 6, pp. 3021–3021, 1995.

[39] K.-J. Bathe, *Finite element method.* Wiley Online Library, 2008.

[40] W. S. Hall, "Boundary element method," *The Boundary Element Method*, pp. 61–83, 1994.

[41] E. A. Navarro, J. Segura, and R. Sanchis, "Aplicación del método de diferencias finitas en el dominio del tiempo (fdtd) al estudio de problemas acústicos bidimensionales."

[42] L. Savioja and U. P. Svensson, "Overview of geometrical room acoustic modeling techniques," *The Journal of the Acoustical Society of America*, vol. 138, no. 2, pp. 708–730, 2015.

[43] J. B. Allen and D. A. Berkley, "Image method for efficiently simulating small-room acoustics," *The Journal of the Acoustical Society of America*, vol. 65, no. 4, pp. 943–950, 1979.

[44] J. Borish, "Extension of the image model to arbitrary polyhedra," *The Journal of the Acoustical Society of America*, vol. 75, no. 6, pp. 1827–1836, 1984.

[45] A. Krokstad, S. Strom, and S. Sørsdal, "Calculating the acoustical room response by the use of a ray tracing technique," *Journal of Sound and Vibration*, vol. 8, no. 1, pp. 118–125, 1968.

[46] S. Laine, S. Siltanen, T. Lokki, and L. Savioja, "Accelerated beam tracing algorithm," *Applied Acoustics*, vol. 70, no. 1, pp. 172–181, 2009.

[47] V. R. Lorenzo, "PFC - Diseño, Implementación y Evaluación de herramientas de simulación y entornos acústicos reverberantes: aplicación a sistemas de reconocimiento automático de habla," Ph.D. dissertation, UPM, 2006.

[48] D. Alonso, "Estado del arte sobre localización, seguimiento y estimación de pose de múltiples locutores usando fusión audiovisual," pp. 1–205, 06/2009 2009.

[49] A. Waibel, R. Stiefelhagen, R. Carlson, J. Casas, J. Kleindienst, L. Lamel, O. Lanz, D. Mostefa, M. Omologo, F. Pianesi, L. Polymenakos, G. Potamianos, J. Soldatos, G. Sutschet, and J. Terken, *Computers in the Human Interaction Loop.* Boston, MA: Springer, 2010, p. 1071–1116.

[50] P. K. Ray and A. Mahajan, "A genetic algorithm-based approach to calculate the optimal configuration of ultrasonic sensors in a 3d position estimation system," *Robotics and Autonomous Systems*, vol. 41, no. 4, pp. 165–177, 2002.

[51] J. O. Roa, A. R. Jiménez, F. Seco, C. Prieto, J. Ealo, and F. Ramos, "Primeros resultados en la optimización de la ubicación de balizas para localización utilizando algoritmos genéticos," *XXVI Jornadas de Automática*, pp. 75–86, 2005.

[52] J. Roa, A. Jiménez, F. Seco, C. Prieto, J. Ealo, F. Ramos, and J. Guevara, "Un método heurıstico mejorado basado en algoritmos genéticos para optimizar la ubicación de balizas en sistemas de localización," *XXVII Jornadas de Automática*, pp. 120–129, 2006.

[53] J. O. Roa, A. R. Jiménez, F. Seco, J. C. Prieto, and J. Ealo, "Optimal placement of sensors for trilateration: Regular lattices vs meta-heuristic solutions," in *International Conference on Computer Aided Systems Theory.* Springer, 2007, pp. 780–787.

[54] D. B. Jourdan and O. L. de Weck, "Layout optimization for a wireless sensor network using a multi-objective genetic algorithm," in *Vehicular technology conference, 2004. VTC 2004-Spring. 2004 IEEE 59th*, vol. 5. IEEE, 2004, pp. 2466–2470.

[55] A. Montazeri, J. Poshtan, and M. Kahaei, "Optimal placement of loudspeakers and microphones in an enclosure using genetic algorithm," in *Control Applications, 2003. CCA 2003. Proceedings of 2003 IEEE Conference on*, vol. 1. IEEE, 2003, pp. 135–139.

[56] Z. Li, K. F. C. Yiu, and Z. Feng, "A hybrid descent method with genetic algorithm for microphone array placement design," *Applied Soft Computing*, vol. 13, no. 3, pp. 1486–1490, 2013.

[57] Z. Diamantis, D. Tsahalis, and I. Borchers, "Optimization of an active noise control system inside an aircraft, based on the simultaneous optimal positioning of microphones and speakers, with the use of a genetic algorithm," *Computational optimization and applications*, vol. 23, no. 1, pp. 65–76, 2002.

[58] B.-T. Wang, "Optimal placement of microphones and piezoelectric transducer actuators for far-field sound radiation control," *The Journal of the Acoustical Society of America*, vol. 99, no. 5, pp. 2975–2984, 1996.

[59] D. Ayllón, R. Gil-Pita, M. Utrilla-Manso, and M. Rosa-Zurera, "An evolutionary algorithm to optimize the microphone array configuration for speech acquisition in vehicles," *Eng. Appl. Artif. Intell.*, vol. 34, pp. 37–44, Sep. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.engappai.2014.05.006

[60] J. D. Schaffer, "Some experiments in machine learning using vector evaluated genetic algorithms," Vanderbilt Univ., Nashville, TN (USA), Tech. Rep., 1985.

[61] ——, "Multiple objective optimization with vector evaluated genetic algorithm," in *Proceeding of the First International Conference of Genetic Algorithms and Their Application*, 1985, pp. 93–100.

[62] P. Hajela and C.-Y. Lin, "Genetic search strategies in multicriterion optimal design," *Structural optimization*, vol. 4, no. 2, pp. 99–107, 1992.

[63] T. Murata and H. Ishibuchi, "Moga: Multi-objective genetic algorithms," in *Evolutionary Computation, 1995., IEEE International Conference on*, vol. 1. IEEE, 1995, p. 289.

[64] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary computation*, vol. 2, no. 3, pp. 221–248, 1994.

[65] J. Knowles and D. Corne, "The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 1. IEEE, 1999, pp. 98–105.

[66] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[67] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," 2001.

[68] D. Engwirda, "Fast points-in-polygon test," MathWorks, 2007. [Online]. Available: https://es.mathworks.com/matlabcentral/fileexchange/10391-fast-points-in-polygon-test

Apéndice A

Funciones y clases desarrolladas

A.1 Introducción

En este anexo se explican las clases desarrolladas en este trabajo, así como sus métodos y las funciones externas que se necesitan para su funcionamiento.

Este trabajo parte del sistema de posicionamiento óptimo de sensores y de arrays de micrófonos para la localización de personas u objetos desarrollado en [15] y [16]

Por tanto en los siguientes apartados, y con el fin de que el contenido de éstos sea completamente autocontenido, se definirán las nuevas funciones creadas o se modificarán las existentes manteniendo las generadas previamente.

A.2 Definición de clases

Durante la ejecución de la GUI se generan las siguientes clases de datos:

- room (Tabla A.1): Contiene la información de la sala. A su vez ésta contiene otras estructuras y cell arrays:
  - surface3D (Tabla A.3): Contiene la información de las paredes en el espacio tridimensional de la sala.
  - surface2D (Tabla A.4): Contiene la información de las paredes en el desarrollo en bidimensional de la sala.
  - obstacles (Tabla A.2): Contiene las distintas estructuras de los obstáculos.
  - MicArray (Tabla A.5): Contiene las medidas de los arrays de micrófonos.
  - sensorPosition (Tabla A.6): Contiene las posiciones posibles de los sensores.
- result (Tabla A.7): Contiene la solución de la ubicación de los sensores obtenida por algoritmo genético. Contiene además estas estructuras:
  - Room (Tabla A.1)
  - GAoptions (Tabla A.8): Contiene las opciones de entrada del algoritmo genético.
  - GAoutput (Tabla A.9): Contiene la solución del algoritmo genético.

- GA (Tabla A.10): Contiene todos los datos sobre el algortimo. A su vez ésta contiene otras estructuras y cell arrays:

  - result (Tabla A.7)
  - options (Tabla A.11): Contiene las opciones del algoritmo genético.
  - output (Tabla A.12): Contiene la estructura de solución del algortimo genético.
  - crossover_options (Tabla A.13): Contiene las opciones de cruce del algortimo genético.
  - mutation_options (Tabla A.14): Contiene las opciones de mutación del algoritmo genético.
  - selectiontype (Tabla A.15): Contiene las opciones de selección del algoritmo genético.

Tabla A.1: Parámetros de la clase *classroom*

Parámetro	**Tamaño**	**Función**
comment	1xn char	Comentario o nombre de la sala
numVertexes	1x1 double	Número de vértices en el suelo
roomFloorHeight	1x1 double	Componente Z del suelo
roomCeilingHeight	1x1 double	Componente Z del techo
Vertexes	(numVertexes)x3 double	Componentes [x,y,z] de los vértices del suelo en metros
Xsize	1x1 double	Xmax-Xmin
Ysize	1x1 double	Ymax-Ymin
Surfaces	1x(num_surfaces) struct	Array de superficies
Grid	(sizegrid)x2	Ubicaciones del grid donde pueden ubicarse los sensores
surface3D	1x1 struct	Información 3D de la sala
surface2D	1x1 struct	Información 2D de la sala
obstacles	1x(num_obstacles) cell	Cell Array de obstáculos
adjwall	1x(nwalls) cell	Paredes completas
ignorewalls	1x(ignoredwalls) double	Paredes completas ignoradas
obstacleoptions	1x1 double	Opciones posibles de análisis de obstáculos
MicArray	1x1 struct	Información del array de micrófonos
sensorPosition	1x1 struct	Restricciones de ubicación de los sensores en la sala
minHeight	1x1 double	Altura mínima para ubicar los sensores
maxHeight	1x1 double	Altura máxima para ubicar los sensores
loaded_population	(population_size)x (2xnum_mics) double	Ubicación de los micrófonos de toda la población

Tabla A.2: Parámetros de la estructura obstacles

Parámetro	Tamaño	Función
vertexes	(nvertexes)x3 double	Contiene las coordenadas [x,y,z] de los vértices del obstáculo
FloorHeight	1x1 double	Contiene la altura mínima del obstáculo
CeilingHeight	1x1 double	Contiene la altura máxima del obstáculo
Fixed	1x1 double	1 o 0: Obstáculo Fijo o no
Name	1xn char	Nombre del obstáculo

Tabla A.3: Parámetros de la estructura surface3D

Parámetro	Tamaño	Función
walls	(nwalls)x(nvertexes)x3 double	Puntos 3D de los vértices de las paredes
floor	(nvertexes)x3 double	Puntos 3D de los vértices del suelo
ceiling	(nvertixes)x3	Puntos 3D de los vértices del techo

Tabla A.4: Parámetros de la estructura surface2D

Parámetro	Tamaño	Función
walls	(nwalls)x(nvertexes)x2 double	Puntos 2D de los vértices de las paredes
floor	(nvertixes)x2 double	Puntos 2D de los vértices del suelo
ceiling	(nvertixes)x2 double	Puntos 2D de los vértices del techo

Tabla A.5: Parámetros de la estructura MicArray

Parámetro	Tamaño	Función
horizdist	1x1 double	Separación horizontal de los micrófonos del array
vertdist	1x1 double	Separación vertical de los micrófonos del array

Tabla A.6: Parámetros de la estructura sensorPosition

Parámetro	Tamaño	Función
walls	1x1 double	1 o 0: Posibilidad o no de sensores en las paredes
floor	1x1 double	1 o 0: Posibilidad o no de sensores en el suelo
ceiling	1x1 double	1 o 0: Posibilidad o no de sensores en el techo

Tabla A.7: Parámetros de la clase classresult

Parámetro	**Tamaño**	**Función**
Room	1x1 classroom	Sala donde se ha ejecutado el algoritmo
n_sensors	1x1 double	Número de sensores
Step	1x1 double	Separación de la rejilla
Matrix_type	1x1 double	Tipo de sensor escogido
Objective_type	1x1 double	Tipo de objetivo seleccionado
Objective_type_2	1x1 double	Tipo de objetivo seleccionado (multiobjetivo)
GAoptions	1x1 struct	Opciones del algoritmo genético
Rebounds	1x1 double	Número de rebotes analizados
xMatrix	(n_sensors)x3 double	Posición de los sensores
Rebounds_matrix	1x1 struct	Matriz de rebotes
Targets	(n_targets)x3 double	Posiciones de los objetivos
Blind_spots	(n_targets)x(n_reb)x(n_surf) cell	Zonas ciegas para cada objetivo
Conv	1x1 cell	Combinaciones de rebotes
GAoutput	1x1 struct	Salida del algoritmo genético
telapsed	1x1 double	Tiempo transcurrido
lastpopulation	(population)x(2xn_sensors) double	Última solución
TestSol	1x1 struct	Coordenadas de los objetivos
SRP	XxYx(n_targets) double	Mapa SRP de la solución
MaxError	1x1) double	Error máximo
MinError	1x1) double	Error mínimo
MeanError	1x1) double	Error medio

Tabla A.8: Parámetros de la estructura GAoptions

Parámetro	**Tamaño**	**Función**
generations	1x1 double	Número de generaciones
populationsize	1x1 double	Tamaño de la población
Functions	1x1 double	Modifica el tipo de cruce y mutación
condition	1x1 double	Modifica el tipo de terminación del algoritmo

Tabla A.9: Parámetros fundamentales de la estructura GAoutput

Parámetro	**Tamaño**	**Función**
generations	1x1 double	Número de generaciones alcanzadas
message	1xn char	Mensaje de finalización

Tabla A.10: Parámetros de la clase classGA

Parámetro	Tamaño	Función
result	1x1 classresult	Valores del algoritmo y resultado
options	1x1 struct	Opciones del algoritmo
size_ind	1x1 double	Tamaño de cada individuo
population	populationsizexsize double	Solución final (o última solución generada)
evaluation	populationsizex1	Valor fitness de population
obj_inter	generationsx1 double	Valor fitness de de soluciones intermedias
sol_inter	generationsxsize_ind double	Soluciones intermedias
t_inter	generationsx1 double	Tiempo empleado para las soluciones intermedias
name_file	1xn char	Nombre del fichero de solución
roundvalue	1x1 double	Decimales de redondeo para la solución
F	1x1 struct	Frentes de pareto (multiobjetivo)
output	1x1 struct	Mensajes de salida
crossoveroptions	1x1 struct	Opciones de cruce
mutationoveroptions	1x1 struct	Opciones de mutación
selectiontype	1x1 struct	Opciones de selección

Tabla A.11: Parámetros fundamentales de la estructura options

Parámetro	Tamaño	Función
population_size	1x1 double	Tamaño de la población
generations	1x1 double	Número de generaciones

Tabla A.12: Parámetros fundamentales de la estructura output

Parámetro	Tamaño	Función
message	1xn char	Mensaje de finalización
generations	1x1 double	Número de generaciones
time	1x1 double	Tiempo empleado
continue	1xn char	'yes'/'no'

Tabla A.13: Parámetros fundamentales de la estructura crossoveroptions

Parámetro	Tamaño	Función
percent	1x1 double	Porcentaje de cruce
function	1xn char	Tipo de función de cruce
points	1x1 double	Número de puntos de cruce
correction	1xn char	'yes'/'no'

Tabla A.14: Parámetros fundamentales de la estructura mutationoveroptions

Parámetro	Tamaño	Función
percent	1x1 double	Porcentaje de cruce
function	1xn char	Tipo de función de mutación
points	1x1 double	Número de puntos de mutación
correction	1xn char	'yes'/'no'
type	1xn char	Tipo de mutación

Tabla A.15: Parámetros fundamentales de la estructura selectiontype

Parámetro	Tamaño	Función
type	1xn char	Tipo de función de selección

A.3 Declaración de funciones

En las Tablas A.16 y A.17 se pueden ver las distintas funciones creadas y/o utilizadas en este trabajo.

Tabla A.16: Funciones creadas y/o utilizadas en este trabajo. Parte 1

Declaración de la función	Uso de la función
conv = calc_conv(nwalls, nrebounds)	Calcula las posibles combinaciones para un número de paredes y un número de rebotes
[cn,on] = inpoly(p, node, edge, TOL)[68]	Función inpolygon rápida
varargout = Interface(varargin)	Define las funciones de la GUI y sirve para lanzarla
mJacob = jacobiano(sensors, target, room)	Calcula la matriz jacobiana. Adaptada de [25]
[t, pairs] = location_to_timedelay (geometry, pairs, locations, fs, c)	Calcula los retardos de localización
y = pair_distance(geometry, pairs)	Calcula la distancia entre pares de micrófonos
point3D=point2Dto3D(point, room)	Permite pasar de un punto del desarrollo 2D a un punto en el espacio 3D NO NECESARIA
point2D=point3Dto2D(point, room)	Permite pasar de un punto en el espacio a un punto del desarrollo 2D

Las funciones señaladas como *NO NECESARIA* no son necesarias para el funcionamiento correcto del sistema pero son útiles para realizar comprobaciones.

Tabla A.17: Funciones creadas y/o utilizadas en este trabajo. Parte 2

Declaración de la función	Uso de la función
Patron_SRP=SRPpattern_cuboidROOM (super_grid,MICS,pares, N_rebotes, Velocidad, Fo, Fs,N_FFT)	Calcula el patrón SRP[17]
objective = traceCRLB(anchors, anchorRef, targets, Matrix_Type,Objective_Type, Functions, rebounds)	Función objetivo del algoritmo Adaptada de [25]
positioning(GA)	Función principal del algortimo genético

A continuación de la Tabla A.18 a la A.27 se detalla para cada una de las funciones los párametros de entrada-salida (incluidas sus carácteristicas) y su funcionamiento interno.

Tabla A.18: $function\ conv = calc\_conv(nwalls,\ nrebounds)$

Parámetro	Entrada/Salida	Información	Tamaño
nwalls	Entrada	Número de paredes con las que calcular los rebotes	1x1 double
nrebounds	Entrada	Número de rebotes	1x1 double
conv	Salida	Posibles combinaciones de rebotes	1x1 cell

Tabla A.19: $function\ [cn,\ on] = inpoly(p,\ node,\ edge,\ TOL)$

Parámetro	Entrada/Salida	Información	Tamaño
p	Entrada	Puntos a comprobar	Nx2 double
node	Entrada	Vértices del polígono	Mx2 double
edge	Entrada	Conexiones entre los vértices	Mx2 double
TOL	Entrada	Tolerancia de la comprobación	1x1 double
cn	Salida	Puntos dentro de la superficie	Nx1 double
on	Salida	Puntos sobre el contorno de la superficie	Nx1 double

Tabla A.20: $function\ mJacob = jacobiano(sensors,\ anchorRef,\ target,\ room)$

Parámetro	Entrada/Salida	Información	Tamaño
solution	Entrada	Solución a probar	1x(2xsensors) double
sensors	Entrada	Número de sensores	1x1 double
room	Entrada	Estructura con la información de la sala	1x1 struct
Matrix_type	Entrada	Tipo se sensores	1x1 double
valid	Salida	1:Solución válida 0:Solución no válida	1x1 double

Tabla A.21: $function\ [t, pairs] = location\_to\_timedelay(geometry, pairs, locations, fs, c)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
geometry	Entrada	Disposición de los sensores	3xN double
pairs	Entrada	Parejas de micrófonos	(nPairs)x2 double
locations	Entrada	Coordenadas 3D de los puntos objetivo	3x(ntargets) double
fs	Entrada	Frecuencia de muestreo en Hz	1x1 double
c	Entrada	Velocidad del sonido en el aire (m/s)	1x1 double
t	Salida	Tiempos de retardo	(nPairs)x(ntargets)double
pairs	Salida	Parejas de micrófonos	3x(ntargets) double

Tabla A.22: $function\ y = pair\_distance(geometry, pairs)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
geometry	Entrada	Disposición de los sensores	3xN double
pairs	Entrada	Parejas de micrófonos	(nPairs)x2 double
y	Salida	Distancias entre pares	(nPairs)x1 double

Tabla A.23: $function\ point3D = point2Dto3D(point, room)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
point	Entrada	Coordenadas 2D de un punto	1x2 double
room	Entrada	Estructura de información de la sala	1x1 struct
point3D	Salida	Coordenadas 3D de un punto	1x3 double

Tabla A.24: $function\ point2D = point3Dto2D(point, room)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
point	Entrada	Coordenadas 3D de un punto	1x3 double
room	Entrada	Estructura de información de la sala	1x1 struct
point3D	Salida	Coordenadas 2D de un punto	1x2 double

Tabla A.25: $function\ Patron\_SRP = SRPpattern\_cuboidROOM(super\_grid, MICS, pares, N\_rebotes, Velocidad, Fo, Fs, N\_FFT)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
super_grid	Entrada	Rejilla con puntos imagen incluidos	(ntargets)x3x(ncov) double
MICS	Entrada	Disposición de los micrófonos	3xN double
pares	Entrada	Parejas de micrófonos	(nPairs)x2 double
N_rebotes	Entrada	Rebotes para cada combinación	(ncov)x1 double
Velocidad	Entrada	Velocidad del sonido	1x1 double
Fo	Entrada	Frecuencia de corte	1x1 double
Fs	Entrada	Frecuencia de muestreo	1x1 double
N_FFT	Entrada	Número de puntos de la FFT	1x1 double
Patron_SRP	Salida	Solución SRP	(ntargets)x(ntargets) double

Tabla A.26: *function objective = traceCRLB(anchors, anchorRef, targets, Matrix_Type, Objective_Type, Functions, rebounds)*

Parámetro	Entrada/Salida	Información	Tamaño
anchors	Entrada	Posiciones 2D de los sensores	Nx2 double
anchorRef	Entrada	Sensor de referencia	1x1 double
targets	Entrada	Rejilla de puntos objetivo	(ntargets)x3 double
Matrix_type	Entrada	Tipo de sensores	1x1 double
Objective_type	Entrada	Tipo de objetivo	1x1 double
Functions	Entrada	Funciones del AG utilizadas	1x1 double
rebounds	Entrada	Número de rebotes	1x1 double
objective	Salida	Objetivo a minimizar	1x1 double

Tabla A.27: *function positioning(GA)*

Parámetro	Entrada/Salida	Información	Tamaño
GA	Entrada	Clase GA	1x1 classGA

A las funciones descritas previamente se añaden los métodos propios de cada una de las tres clases definidas previamente classroom (Tabla A.28), classresult (Tabla A.29)y classGA (Tabla A.30) definidos de la Tabla A.31 a la A.58.

Tabla A.28: Métodos propios de classroom

Declaración de la función	Uso de la función
room = LoadEnvironment(room, environment_file)	Carga el fichero .srf de la sala
plot_room(room)	Imprime en una figura el contorno de la sala
room = upd_room(room)	Calcula los puntos 3D y 2D de la sala
target = calc_grid_points(room, Step, testHeight)	Calcula la rejilla de la sala
[is_obstacled, intersec_point, Surface] = checkobstacles(room, sensorPosition, targetPoint, complete)	Calcula si una trayectoria
intersec_point = calc_intersec(room, sensorPosition, targetPoint, wall)	Calcula la intersección
[in, on] = is_in_surface(room, point, Surface)	Calcula si un punto está en una superficie
plot_develop(room, walls, floor, ceiling)	Imprime el desarrollo 2D de la sala
[valid, non_valid] = is_valid_solution (room, solution, sensors, Matrix_type)	Comprueba si la solución es válida
[super_grid, N_rebotes, conv] = calculate_IM_grid (room, P, nsurf, rebounds)	Calcula los puntos imagen
[blind_spots = calc_blind_spots(room, super_grid, conv)	Calcula las zonas ciegas de la sala
matrix = Checkpoints(room, mics, super_grid, blind_spots, conv))	Calcula si los puntos estón obstaculizados
valid_path = check_path(room, sensor, targets, walls)	Comprueba si una trayectoria es valida

Tabla A.29: Métodos propios de classresult

Declaración de la función	**Uso de la función**
result = testsolution(result, testgrid, a)	Calcula el error de la solución
printwalls(result)	Imprime las paredes de la sala
result = intermediate_results(result, step, a)	Calcula las soluciones intermedias
save_cluster_script(result, minsensors, maxsensors)	Salva el script para lanzarlo en el cluster
objective = fitness_function(result, anchors)	Calcula la función objetivo
[valid, non_valid] = is_valid_solution(result, solution)	Comprueba si una solución es válida

Tabla A.30: Métodos propios de classGA

Declaración de la función	**Uso de la función**
GA = ga_rmp(GA)	Función del algoritmo desarrollado
children = ga_crossover(GA)	Realiza el proceso de cruce
children = ga_mutation(GA, children)	Realiza el proceso de mutación
GA = ga_selection(GA, children, evaluation)	Realiza el proceso de selección
GA = ga_initialize(GA)	Inicializa el algoritmo
evaluation = ga_evaluation(GA, population_test)	Evalúa la población
GA = ga_create_population(GA)	Crea la población inicial
GA = non_dominated_sort(GA, evaluation)	Calcula los frentes de pareto
GA = crowding_distance(GA, evaluation)	Ordena los frentes de pareto

Tabla A.31: $function\ room = LoadEnvironment(room, environment\_file)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
room	Entrada	Estructura con la información de la sala	1x1 struct
environment_file	Entrada	Nombre del fichero	1xn char
room	Salida	Estructura con la información de la sala	1x1 struct

Tabla A.32: $function\ plot\_room(room)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
room	Entrada	Estructura con la información de la sala	1x1 struct

Tabla A.33: $function\ room = upd\_room(room)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
room	Entrada	Estructura con la información de la sala	1x1 struct
room	Salida	Estructura con la información de la sala	1x1 struct

Tabla A.34: $function\ target = calc\_grid\_points(room, Step, testHeight)$

Parámetro	Entrada/Salida	Información	Tamaño
room	Entrada	Estructura con la información de la sala	1x1 struct
Step	Entrada	Tamaño rejilla puntos objetivo	1x1 double
testHeight	Entrada	Altura de la rejilla	1x1 double
target	Salida	Coordenadas 3D de los puntos objetivo	(ntargets)x3 double

Tabla A.35: $function\ [is\_obstacled, intersec\_point, Surface] = checkobstacles(room, sensorPosition, targetPoint, complete)$

Parámetro	Entrada/Salida	Información	Tamaño
room	Entrada	Estructura con la información de la sala	1x1 struct
sensorPosition	Entrada	Posición del sensor	1x3 double
targetPoint	Entrada	Posición del objetivo	1x3 double
complete	Obtención punto corte	1x1 double	
is_obstacled	Salida	Trayectoria obstaculizada o no	1x1 double
intersec_point	Salida	Punto de intersección	1x3 double
Surface	Salida	Superficie de corte	1x1 double

Tabla A.36: $function\ intersec\_point = calc\_intersec(room, sensorPosition, targetPoint, wall)$

Parámetro	Entrada/Salida	Información	Tamaño
room	Entrada	Estructura con la información de la sala	1x1 struct
sensorPosition	Entrada	Posición del sensor	1x3 double
targetPoint	Entrada	Posición del objetivo	1x3 double
wall	Superficie de intersección	1x1 double	
intersec_point	Salida	Punto de intersección	1x3 double

Tabla A.37: $function\ [in, on] = is\_in\_surface(room, point, Surface)$

Parámetro	Entrada/Salida	Información	Tamaño
room	Entrada	Estructura con la información de la sala	1x1 struct
point	Entrada	Posición del punto	1x3 double
Surface	Entrada	Superficie	1x1 double
in	Salida	Punto en la superficie	1x1 double
on	Salida	Punto sobre la arista	1x1 double

Tabla A.38: *function plotdevelop*$(room, walls, floor, ceiling)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
room	Entrada	Estructura de información de la sala	1x1 struct
walls	Entrada	Indica la posibilidad de colocar los sensores en las paredes	1x1 double
floor	Entrada	Indica la posibilidad de colocar los sensores en el suelo	1x1 double
Ceiling	Entrada	Indica la posibilidad de colocar los sensores en las paredes	1x1 double

Tabla A.39: *function* $[valid, non\_valid] = is\_valid\_solution(room, solution, sensors, Matrix\_type)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
room	Entrada	Estructura con la información de la sala	1x1 struct
solution	Entrada	Solución	(2xn_sensors)x2 double
sensors	Entrada	Posición de los sensores	n_sensorsx2 double
Matrix_type	Entrada	Tipo de array de micrófonos	n_sensorsx2 double
valid	Salida	Solución válida	1x1 double
non_valid	Salida	Posiciones no válidas	1x1 double

Tabla A.40: *function* $[super\_grid, N\_rebotes, conv] = calculate\_IM\_grid(room, P, nsurf, rebounds)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
room	Entrada	Estructura con la información de la sala	1x1 struct
P	Entrada	Puntos de la rejilla	Nx3 double
nsurf	Número de superficies	1x1 double	
rebounds	Número de rebotes	1x1 double	
super_grid	Salida	Rejilla con puntos imagen	Mx3 double
N_rebotes	Salida	Rebotes del punto	Mx1 double
conv	Salida	Combinación de paredes	Mx1 double

Tabla A.41: *function* $blind\_spots = calc\_blind\_spots(room, super\_grid, conv)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
room	Entrada	Estructura con la información de la sala	1x1 struct
super_grid	Entrada	Puntos de la rejilla	Mx3 double
conv	Entrada	Combinación de rebotes	1x1 double
blind_spots	Salida	Zonas ciegas	1x1 cell

Tabla A.42: $function\ matrix = Checkpoints(room, mics, super\_grid, blind\_spots, conv)$

Parámetro	Entrada/Salida	Información	Tamaño
room	Entrada	Estructura con la información de la sala	1x1 struct
mics	Entrada	Micrófonos	n_micsx3 double
super_grid	Entrada	Puntos de la rejilla	Mx3 double
blind_spots	Entrada	Zonas ciegas	1x1 cell
conv	Entrada	Combinación de rebotes	1x1 double
matrix	Salida	Matriz de puntos posibles	Mxn_mics cell

Tabla A.43: $function\ valid\_path = check_path(room, sensor, targets, walls)$

Parámetro	Entrada/Salida	Información	Tamaño
room	Entrada	Estructura con la información de la sala	1x1 struct
sensor	Entrada	Sensor	1x3 double
targets	Entrada	Puntos de la rejilla	Mx3 double
walls	Entrada	Paredes	1xnwalls double
valid_path	Salida	Trayectoria válida o no	1x1 double

Tabla A.44: $function\ result = testsolution(result, testgrid, a)$

Parámetro	Entrada/Salida	Información	Tamaño
result	Entrada	Estructura de la solución	1x1 struct
testgrid	Entrada	Separación del grid	1x1 double
a	Entrada	Tolerancia de error	1x1 double
result	Salida	Estructura de la solución	1x1 struct
result	Entrada	Estructura de la solución	1x1 struct

Tabla A.45: $function\ printwalls(result)$

Parámetro	Entrada/Salida	Información	Tamaño
result	Entrada	Estructura de la solución	1x1 struct

Tabla A.46: $function\ intermediate\_results(result)$

Parámetro	Entrada/Salida	Información	Tamaño
result	Entrada	Estructura de la solución	1x1 struct

Tabla A.47: $function\ save\_cluster\_script(result, minsensors, maxsensors)$

Parámetro	Entrada/Salida	Información	Tamaño
result	Entrada	Estructura de la solución	1x1 struct
minsensors	Entrada	Número mínimo de sensores	1x1 double
maxsensors	Entrada	Número máximo de sensores	1x1 double

Tabla A.48: $function\ objective = fitness\_function(result, anchors)$

Parámetro	Entrada/Salida	Información	Tamaño
result	Entrada	Estructura de la solución	1x1 struct
anchors	Entrada	Posición de los micrófonos	n_micsx3 double
objective	Entrada	Objetivo de la solución	1x(1 ó 2) double

Tabla A.49: $function\ [valid, non\_valid] = is\_valid\_solution(result, solution)$

Parámetro	Entrada/Salida	Información	Tamaño
result	Entrada	Estructura de la solución	1x1 struct
solution	Entrada	Solución a probar	1x(2xsensors) double
valid	Salida	Posiciones válidas	1xN double
non_valid	Salida	Posiciones no válidas	1xN double

Tabla A.50: $function\ GA = ga\_rmp(GA)$

Parámetro	Entrada/Salida	Información	Tamaño
GA	Entrada	Estructura de la solución	1x1 struct
GA	Salida	Estructura de la solución	1x1 struct

Tabla A.51: $function\ children = ga\_crossover(GA)$

Parámetro	Entrada/Salida	Información	Tamaño
GA	Entrada	Estructura de la solución	1x1 struct
children	Salida	Soluciones nuevas	Mx2 double

Tabla A.52: $function\ children = ga\_mutation(GA, children)$

Parámetro	Entrada/Salida	Información	Tamaño
GA	Entrada	Estructura de la solución	1x1 struct
children	Entrada	Elementos con posibilidad de mutación	Mx2 double
children	Salida	Soluciones nuevas	Mx2 double

Tabla A.53: $function\ GA = ga\_selection(GA, children, evaluation)$

Parámetro	Entrada/Salida	Información	Tamaño
GA	Entrada	Estructura de la solución	1x1 struct
children	Entrada	Soluciones nuevas	Mx2 double
evaluation	Entrada	Evaluación de nuevas soluciones	Mx1 double
GA	Salida	Estructura de la solución	1x1 struct

Tabla A.54: $function\ GA = ga\_initialize(GA)$

Parámetro	Entrada/Salida	Información	Tamaño
GA	Entrada	Estructura de la solución	1x1 struct
GA	Salida	Estructura de la solución	1x1 struct

Tabla A.55: $function\ evaluation = ga\_evaluation(GA, population\_test)$

Parámetro	Entrada/Salida	Información	Tamaño
GA	Entrada	Estructura de la solución	1x1 struct
population_test	Entrada	Soluciones a evaluar	Mx2 double
evaluation	Salida	Evaluación de nuevas soluciones	Mx1 double

Tabla A.56: $function\ GA = ga\_create\_population(GA)$

Parámetro	Entrada/Salida	Información	Tamaño
GA	Entrada	Estructura de la solución	1x1 struct
GA	Salida	Estructura de la solución	1x1 struct

Tabla A.57: $function\ GA = non\_dominated\_short(GA, evaluation)$

Parámetro	Entrada/Salida	Información	Tamaño
GA	Entrada	Estructura de la solución	1x1 struct
evaluation	Entrada	Evaluación de la población	Mx1 double
GA	Salida	Estructura de la solución	1x1 struct

Tabla A.58: $function\ GA = crowding\_distance(GA, evaluation)$

Parámetro	Entrada/Salida	Información	Tamaño
GA	Entrada	Estructura de la solución	1x1 struct
evaluation	Entrada	Evaluación de la población	Mx1 double
GA	Salida	Estructura de la solución	1x1 struct

Apéndice B

Manual de usuario

B.1 Introducción

La GUI desarrollada en este TFM para MATLAB y las funciones necesarias para su uso han sido creadas para calcular la posición óptima de un conjunto de sensores en una sala determinada por el usuario, con el fin de calcular la posición del hablante en dicha sala.

B.2 Manual

B.2.1 Instalación

Para el uso de estas funciones es sólo es necesario tener instalado MATLAB con la versión 2012a o posterior en el equipo.

B.2.2 Ejecución Interfaz

Iniciar la GUI a través de MATLAB, abriendo el fichero “Interface.m” y corriendo el código mediante F5 o “Run”, o bien tecleando “Interface” por línea de comandos estando en la carpeta donde se copiaron los ficheros. Lo que arrancará la pantalla de inicio de la interfaz B.1.

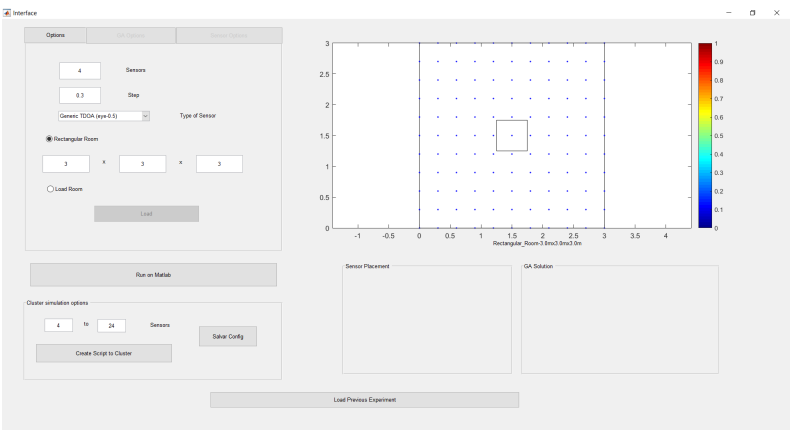


Figura B.1: Interfaz Gráfica

La interfaz de puede subdividir en 6 partes como en B.2: opciones (1), creación de script(2), visualización(3), resultados (4), lanzamiento (5) y mostrado (6).

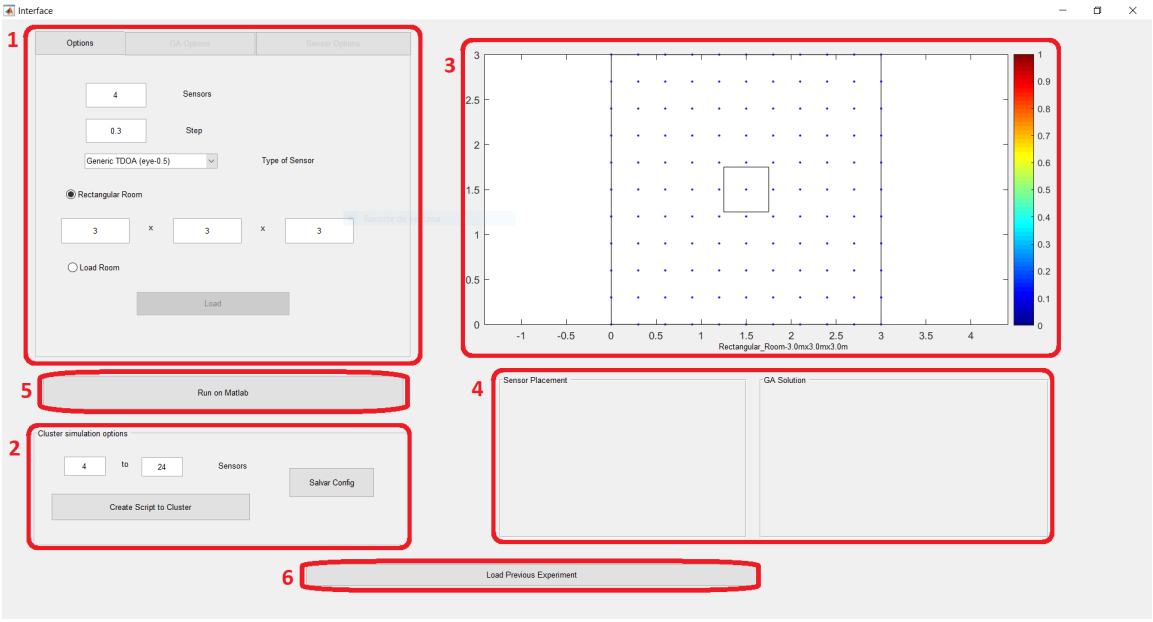


Figura B.2: Partes de la interfaz

B.2.2.1 Opciones

La parte de opciones tiene los siguientes submenús:

1. Submenú Options B.3: Permite cambiar las opciones básicas.

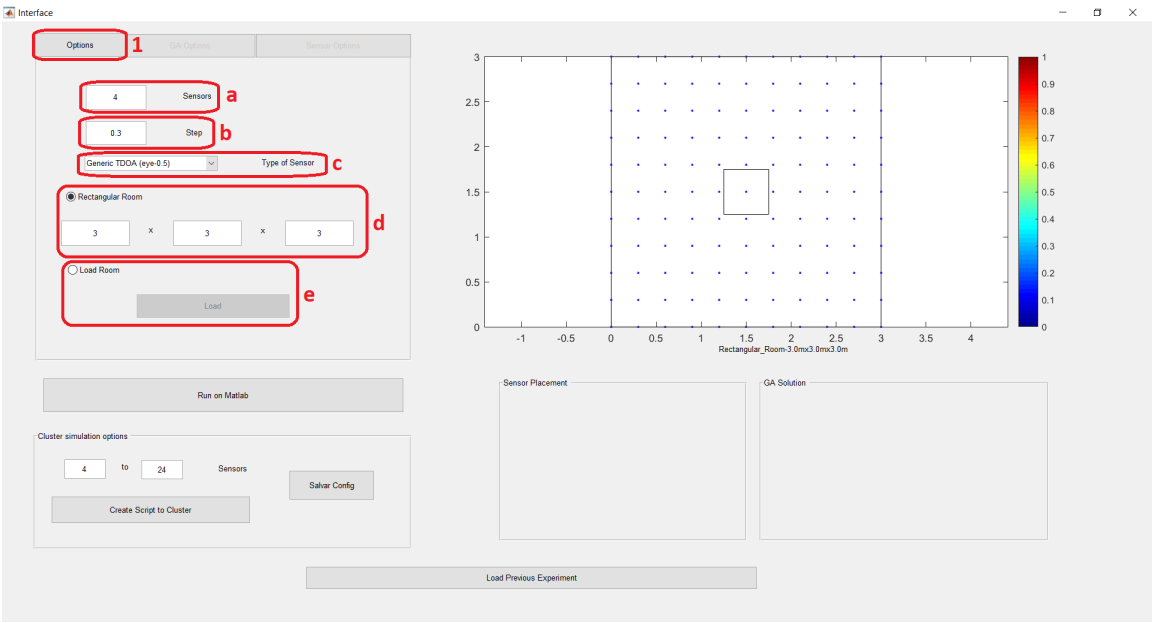


Figura B.3: Opciones del submenú Options

(a) Determina el número de sensores o arrays de micrófonos que se quieren posicionar. Dicho valor está limitado inferiormente a 4 para sensores aislados o a 2 para arrays de micrófonos.

(b) Determina la separación de la rejilla que se va a analizar. Dicho valor esta limitado a 0.1, 0.2 o 0.3 para arrays de micrófonos pero sin limitaciones para sensores aislados.

(c) Sirve para seleccionar el tipo de sensores y el tipo de matriz de covarianza usado.

- Generic TDOA (eye-0.5): Selecciona sensores aislados con la matriz de covarianza B.1, que indica que el error es independiente de la distancia a los sensores.

$$\Sigma = \begin{pmatrix} 1 & 0{,}5 & \cdots & 0{,}5 \\ 0{,}5 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0{,}5 \\ 0{,}5 & \cdots & 0{,}5 & 1 \end{pmatrix} \tag{B.1}$$

- Generic TDOA (sq. dist. dep.): Selecciona sensores aislados con la matriz de covarianza B.2, que indica que el error es dependiente al cuadrado de la distancia a los sensores.

$$\Sigma = \begin{pmatrix} dr^2 + ds_1^2 & dr^2 & \cdots & dr^2 \\ dr^2 & dr^2 + ds_2^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & dr^2 \\ dr^2 & \cdots & dr^2 & dr^2 + ds_{N-1}^2 \end{pmatrix} \tag{B.2}$$

- Isolated Microphones: Selecciona micrófonos como tipo de sensores para la localización basada en SRP.
- Microphone Array(SRP): Selecciona el tipo de sensor como arrays de micrófonos con topología de T invertida con localización basada en SRP, con o sin rebotes.
- Microphone Array(TDOA): Selecciona el tipo de sensor como arrays de micrófonos con topología de T invertida con localización basada en TDOA, sin rebotes.
  Todas la posibilidades cuando son seleccionadas activan o desactivan (según cual esté seleccionada) características de la interfaz que son relevantes o no para este ese de sensores con la localización seleccionada, como la distancia entre los micrófonos, la limitación del número de sensores o la activación o desactivación de la selección de objetivo.

(d) Si está activo selecciona una sala rectangular por defecto un cubo de 3 metros de lado con una columna cuadrada en el centro y establece las dimensiones X, Y y Z de la sala rectangular.

(e) Carga una sala definida en un fichero "*.env*", como el de la Figura B.4, que a su vez llama a un conjunto de superficies definidas en ficheros "*.srf*" , como el de la Figura B.5.

El archivo "*.env*" estará dividido en 3 partes.

i. Una primera parte encabezada por [$EnvironmentInfo$] contiene la información sobre las características generales de la sala:
   - content: Contenido del fichero
   - version: Número de versión.
   - comment: Nombre de la sala.
   - numSurfaces: Número de superficies que forman la sala.

ii. Un segundo apartado encabezado por [$Environment$] contiene la ruta relativa de los ficheros de la sala.

iii. El tercer apartado encabezado por [$SurfaceFiles$] que contiene los nombres de los ficheros "*.srf*" de las superficies que forman la sala, de la siguiente forma: $surfaceN = nombre.srf$. El nombre del fichero además debe contener información relevante para saber

```
[EnvironmentInfo]

content = Environment definition file
version = 1.1
comment = IDIAP Demo room (for av16.3 simulation)

numSurfaces = 7

[Environment]

dirEnvironments = environments/IdiapRoom/

[SurfaceFiles]

surface0 = leftWall.srf
surface1 = rightWall.srf
surface2 = backWall.srf
surface3 = frontWall.srf
surface4 = floor.srf
surface5 = ceiling.srf
surface6 = table.srf
```

Figura B.4: Ejemplo de archivo ".env" de definición de sala

```
[SurfaceInfo]

content = Surface definition file
version = 1.1
comment = Idiap demo room

numVertexes = 4

[Vertexes]

vertex0 = 1800 2200 1670
vertex1 = 1800 −6000 1670
vertex2 = −1800 −6000 1670
vertex3 = −1800 2200 1670
```

Figura B.5: Ejemplo de archivo ".srf" de definición de superficie

de que tipo de superficie se trata, si es una pared (debe contener *wall*), si es el techo (debe contener *ceiling*), si es el suelo (debe contener $floor$), si es una columna o un obstáculo fijo (debe contener *column*), el resto del nombre del fichero y el resto de superficies es independiente, aunque es recomendable denominarlo de forma coherente.

A su vez los archivos "$.srf''$ estarán dividido en 2 partes.

i. Una primera parte encabezada por $[SurfaceInfo]$ contiene la información sobre las características generales de la sala:
   - content: Contenido del fichero
   - version: Número de versión.
   - comment: Nombre de la sala.
   - numVertexes: Número de vértices que forman la sala.

ii. El segundo apartado encabezado por $[Vertexes]$ contiene las coordenadas [x y z] de cada vértice de la superficie., de la siguiente forma: $vertexN = xyz$

2. Submenú GA Options B.6: Permite modificar los parámetros del Algoritmo Genético

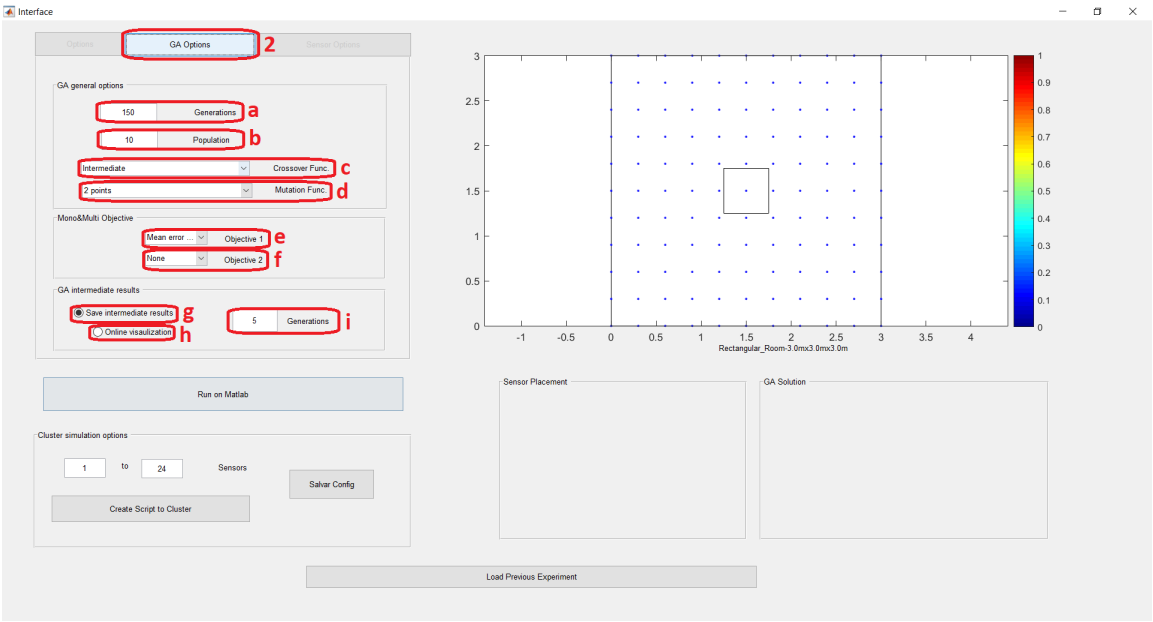


Figura B.6: Opciones del submenú GA Options

(a) Determina el número máximo de generaciones que alcanzará el algoritmo genético.

(b) Fija el tamaño de la población del algoritmo genético.

(c) Selecciona la función de Cruce utilizada por el algoritmo genético, que puede escogerse entre:

- Intermediate: Cruce por punto intermedio.
- 1 point: Cruce por un punto.
- 2 points: Cruce por dos puntos.

(d) Selecciona la función de Mutación utilizada por el algoritmo genético, que puede escogerse entre:

- 1 point: Mutación en un punto.
- 2 points: Mutación en dos puntos.

⋮

- 8 points: Mutación en ocho puntos.

(e) Define el primer (y único para el algoritmo mono-objetivo) objetivo entre los siguientes:

Para SRP:

- Mean error: Minimiza el error medio para todos los puntos de la sala.
- Max error: Minimiza el error máximo de todos los puntos de la sala.
- (Max-Min) error: Minimiza la diferencia entre el error máximo y el mínimo de todos los puntos de la sala.

Para TDOA:

- Trace: Minimiza el valor medio de la Traza de la CRLB.
- Determinant: Minimiza el valor del determinante de la Traza de la CRLB.
- Max Trace: Minimiza el valor máximo de la Traza de la CRLB.

(f) Define el segundo objetivo entre los siguientes:

Para SRP:

- None: ningún objetivo (selecciona algoritmo mono-objetivo).
- Mean error: Minimiza el error medio para todos los puntos de la sala.
- Max error: Minimiza el error máximo de todos los puntos de la sala.
- (Max-Min) error: Minimiza la diferencia entre el error máximo y el mínimo de todos los puntos de la sala.

Para TDOA:

- None: ningún objetivo (selecciona algoritmo mono-objetivo).
- Trace: Minimiza el valor medio de la Traza de la CRLB.
- Determinant: Minimiza el valor del determinante de la Traza de la CRLB.
- Max Trace: Minimiza el valor máximo de la Traza de la CRLB.

(g) Define si se desean guardar soluciones intermedias o no.

(h) Define si además de guardar las soluciones se desea que se visualicen.

(i) Define el intervalo de generaciones para guardar/visualizar las soluciones.

3. Submenú Sensor Options B.7: Permite modificar los sensores.

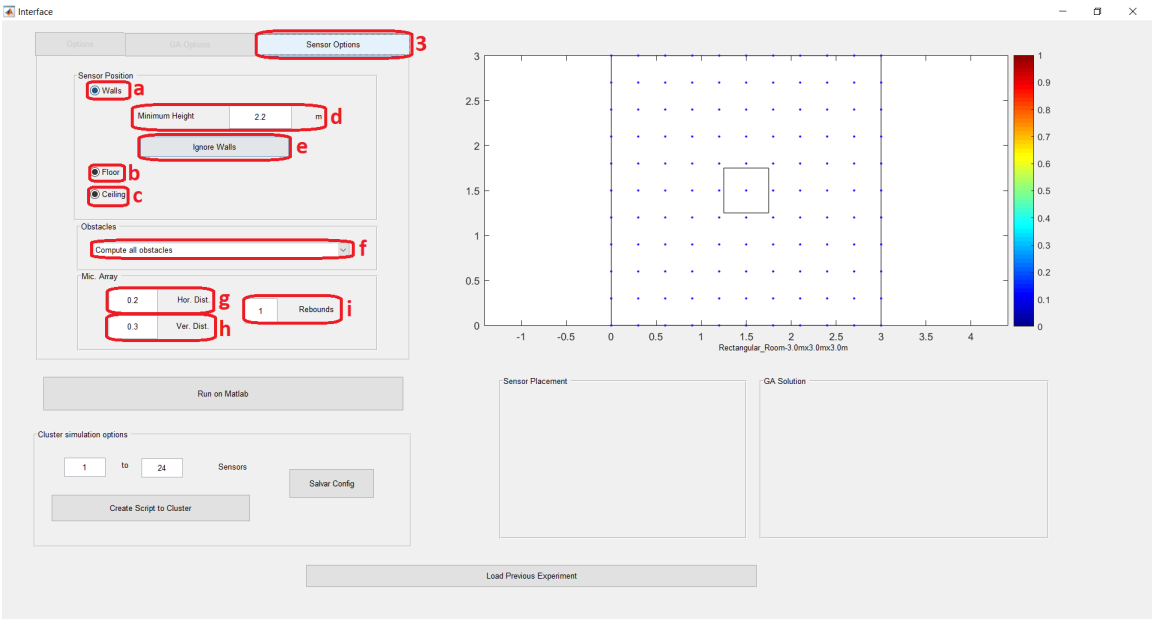


Figura B.7: Opciones del submenú Sensor Options

(a) Si esta seleccionado permite la ubicación de los sensores en las paredes.

(b) Si está seleccionado permite la ubicación de los sensores en el suelo.

(c) Si está seleccionado permite la ubicación de los sensores en el techo.

(d) Determina la altura mínima de los sensores en las paredes.

(e) Permite seleccionar las paredes donde no se pueden ubicar los sensores, al presionarlo aparecerá una cruceta para pinchar sobre la pared a ignorar (selecciona la más cercana a donde se pinche).

(f) Determina si se tienen en cuenta o no los obstáculos para calcular la solución y cuales se pintan, a escoger entre:

- Compute all obstacles: Calcula el objetivo con todos los obstáculos.

- Ignore non fixed obstacles: Ignora para el calculo del objetivo los obstáculos móviles (p.e. mesas).
- Ignore and non plot non fixed obstacles: Ignora para el calculo del objetivo los obstáculos móviles (p.e. mesas), además no las representa.
- Ignore all obstacles: Ignora todos los obstáculos para el cálculo del objetivo y solo se representan los obstáculos fijos.
- Ignore and non plot all obstacles: Ignora todos los obstáculos para el cálculo del objetivo y no representa ninguno.

(g) Está desactivado para sensores aislados y sirve para determinar la separación vertical entre micrófonos.

(h) Está desactivado para sensores aislados y sirve para determinar la separación vertical entre micrófonos.

(i) Está desactivado para sensores genéricos y determina número de rebotes que se tienen en cuenta para el cálculo.

B.2.2.2 Cluster

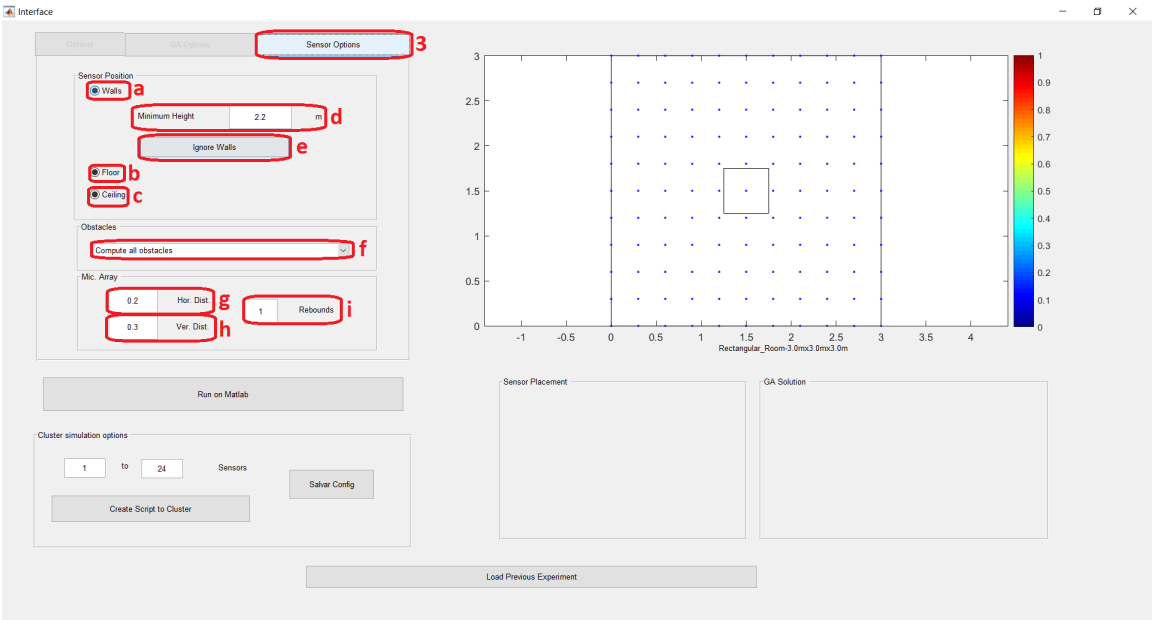


Figura B.8: Opciones del Cluster

1. Permite crear un script ".bash" ejecutable en el cluster para realizar la misma prueba con un distinto número de sensores de forma más rápida. Como el de la Figura B.9.

2. Permite salvar la configuración de la sala para experimentos en el cluster o en el PC.

B.2.2.3 Visualización

La parte de visualización tiene dos cometidos.

El primero es el de visualizar cómo es la sala que vamos a tener en cuenta, los obstáculos que se tienen o no en cuenta y el tamaño de la rejilla. Como los ejemplos de B.10.

```
#!/bin/bash

# General definitions
MATLAB_PROGRAM=test

SWEEP_Step="0.30"
SWEEP_Matrix_Type="3"
SWEEP_n_sensors="3 4 5"
SWEEP_room="_ISPACE_room_definition_.mat"
SWEEP_Objective_Type="1"
SWEEP_Generations='200'
SWEEP_Population='50'
SWEEP_Functions='2'

Step=$SWEEP_Step
room=$SWEEP_room
Objective_Type=$SWEEP_Objective_Type
Generations=$SWEEP_Generations
Population=$SWEEP_Population
Functions=$SWEEP_Functions

for Matrix_Type in $SWEEP_Matrix_Type
do
        for n_sensors in $SWEEP_n_sensors
                do
                        echo "--------------------------------------"
                        echo "Submitting $MATLAB_PROGRAM for n_sensors=$n_sensors"
                        geintra_run_matlab $MATLAB_PROGRAM "Step=$Step;
                        room=$room;Matrix_Type=$Matrix_Type;n_sensors=$n_sensors;
                        Generations=$Generations;Population=$Population;
                        Objective_Type=$Objective_Type;Functions=$Functions"
                done
        done
done
```

Figura B.9: Ejemplo de archivo ".bash" de script para el cluster

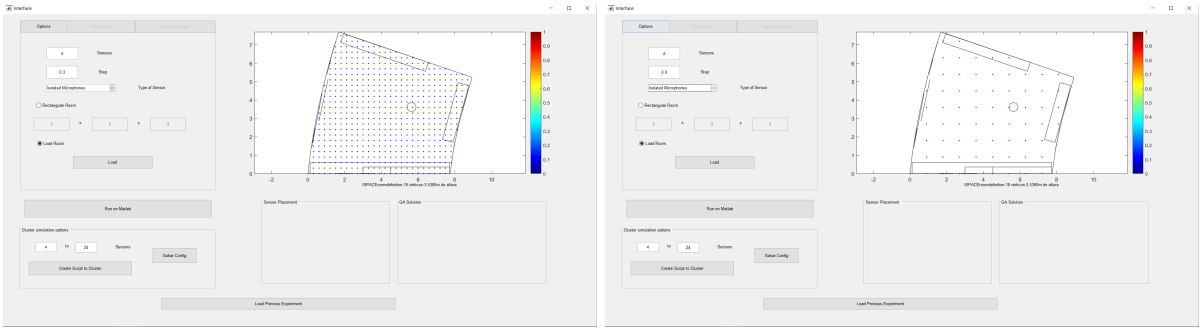


Figura B.10: Distintas rejillas para sala ISPACE

El segundo cometido es el de visualizar el resultado en dos dimensiones como en el de la Figura B.13. Dicho resultados también se ve en 3D en uno de los pop-up que aparecen cuando se finaliza la ejecución o cuando se carga un resultado anterior, como en la Figura B.11, así como el la mayoría de las opciones de configuración aparecen en otro pop-up, como el de la Figura B.12.

B.2.2.4 Resultados

La parte de resultado se divide en dos:

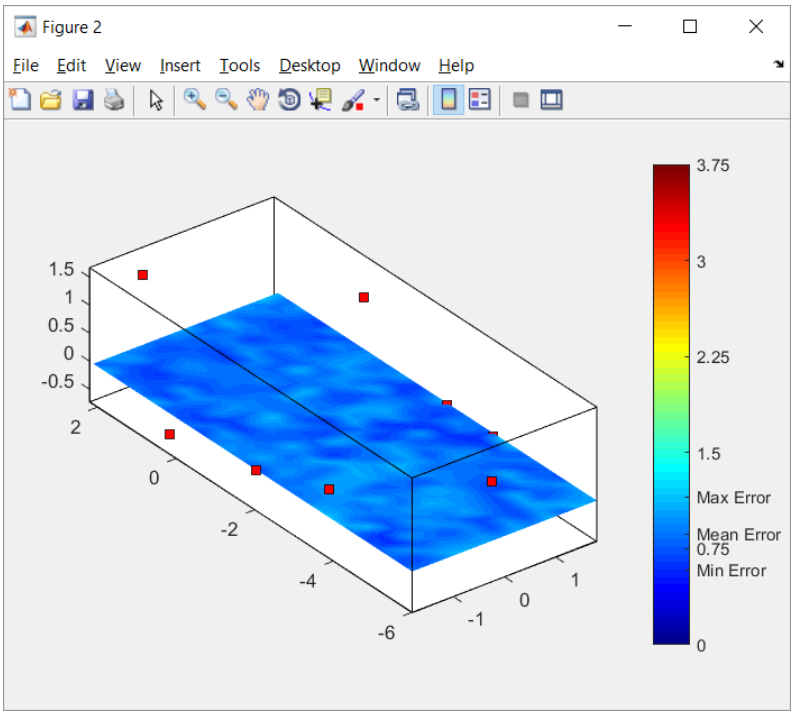


Figura B.11: Resultado 3D

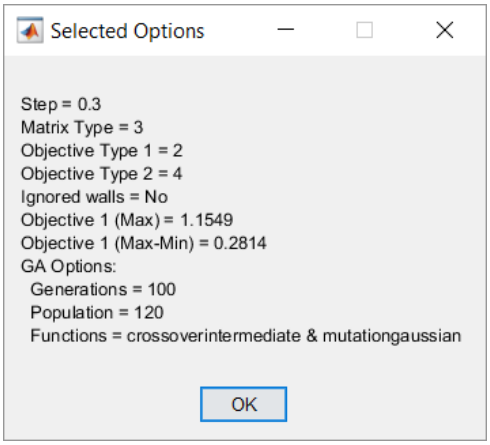


Figura B.12: Opciones del resultado

1. Aporta la posición de los sensores (en el caso de los arrays de micrófonos nos aporta la posición del micrófono central).

2. Aporta la solución del algoritmo genético. En la Figura B.13 se puede ver un ejemplo de resultado.

3. Permite visualizar el patrón SRP para un punto de la sala, seleccionando el punto como el de la Figura B.14

B.2.2.5 Lanzamiento de algoritmo en la interfaz

El botón *Run on Matlab* permite lanzar el algoritmo genético con los parámetros seleccionados en la interfaz, con la posibilidad de la visualización en tiempo real.

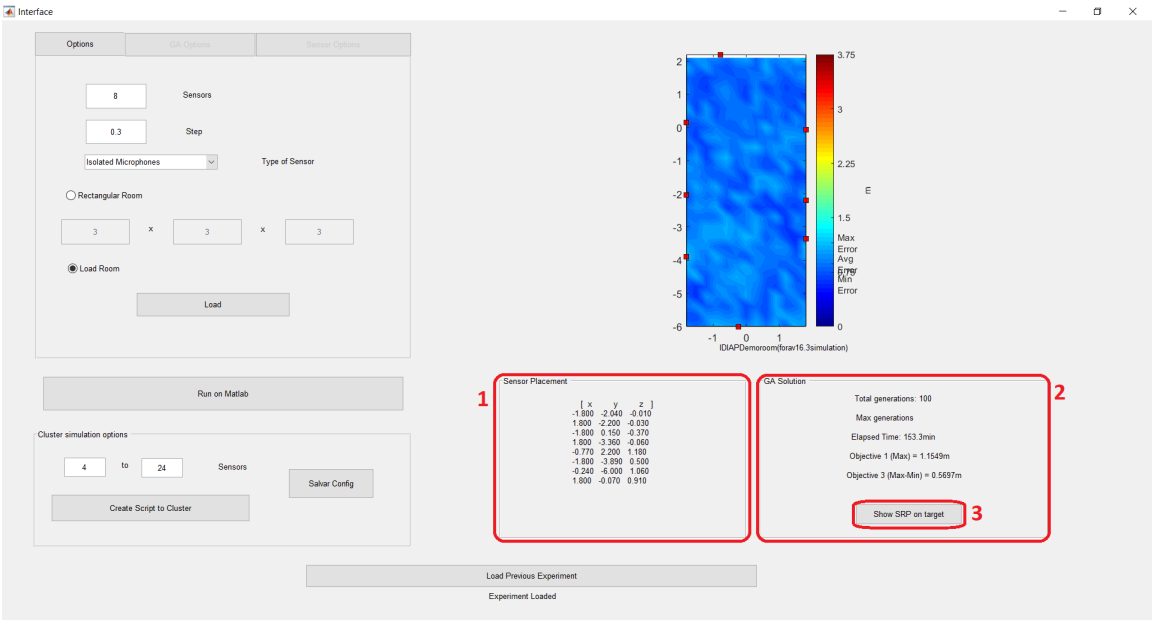


Figura B.13: Resultados

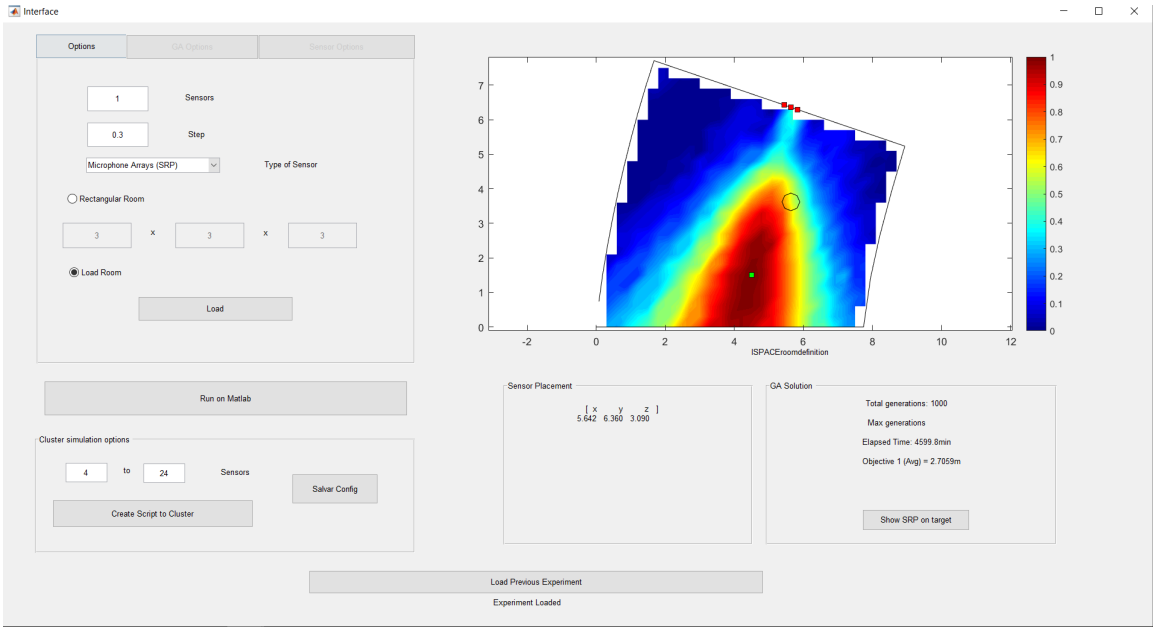


Figura B.14: SRP para un punto de la sala ISPACE

B.2.2.6 Mostrado de resultados anteriores

La interfaz también permite visualizar resultados anteriores, para ello de forma automática guarda en un fichero cuyo nombre contiene la información de la sala, y parámetros importantes de las opciones.

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá